3/5/1 (Item 1 from file: 351)
DIALOG(R)File 351:DERWENT WPI

012231289 **Image available**
WPI Acc No: 99-037396/199904
XRPX Acc No: N99-028236
Hash value generating method for data encryption or decryption e.g. for electronic mail - generating hash valves, keys and cipher text rapidly, and when message is sent inputting divided data of message and applying injection extension processing to obtain longer data length
Patent Assignee: HITACHI LTD (HITA)
Inventor: KURUMATANI H; TAKARAGI K
Number of Countries: 026 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Main IPC	Week
EP 886399	A2	19981223	EP 98110182	A	19980604	H04L-009/32	199904 B
JP 10340048	A	19981222	JP 97149423	A	19970606	G09C-001/00	199910

Priority Applications (No Type Date): JP 97149423 A 19970606
Cited Patents: No-SR.Pub
Patent Details:

Patent	Kind	Lan	Pg	Filing Notes	Application	Patent
EP 886399	A2	E	57			
Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI						
JP 10340048	A		53			

Abstract (Basic): EP 886399 A
The hash value generating method for digital signature or data encryption involves dividing target data into at least two blocks. Character-substitution and/or transposition processing is then performed on any one of the blocks. The processed data is then subjected to a multiplication process so that the data product is longer than the original data length.
The product data is then divided into at least two blocks, and character-substitution and/or transposition processing is again performed on any one of the blocks. The processing of the data may involve an injection extension transformation in which an output value is absolutely different if an input value is different (injection) and the length of the output value is longer than the length of the input value (extension).
USE - Data scrambling e.g. in computer network, e.g. for electronic mail system.
ADVANTAGE - Message is converted to hash value which is difficult to inverse convert.
Dwg.1/27
Title Terms: HASH; VALUE; GENERATE; METHOD; DATA; ENCRYPTION; DECRYPTER; ELECTRONIC; MAIL; GENERATE; HASH; VALVE; KEY; CIPHER; TEXT; RAPID; MESSAGE; SEND; INPUT; DIVIDE; DATA; MESSAGE; APPLY; INJECTION; EXTEND; PROCESS; OBTAIN; LONG; DATA; LENGTH
Derwent Class: W01
International Patent Class (Main): G09C-001/00; H04L-009/32
International Patent Class (Additional): H04L-009/06; H04L-009/30
File Segment: EPI

3/5/2 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

06056948 **Image available**
HASH VALUE GENERATING METHOD, DATA CIPHERING METHOD, DATA DECIPHERING METHOD, HASH VALUE GENERATING DEVICE DATA CIPHERING DEVICE, AND DATA DECIPHERING DEVICE

PUB. NO.: 10-340048 A]
PUBLISHED: December 22, 1998 (19981222)
INVENTOR(s): TAKARAGI KAZUO
KURUMAYA HIROYUKI
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 09-149423 [JP 97149423]
FILED: June 06, 1997 (19970606)
INTL CLASS: [6] G09C-001/00; G09C-001/00; G09C-001/00; H04L-009/06; H04L-009/30
JAPIO CLASS: 44.9 (COMMUNICATION -- Other); 44.3 (COMMUNICATION -- Telegraphy)
JAPIO KEYWORD: R131 (INFORMATION PROCESSING -- Microcomputers & Microprocessers); R303

ABSTRACT

PROBLEM TO BE SOLVED: To disorder data efficiently by performing a multiplying process which makes the length of an output value longer than that of an input value in a hash value generating process.

SOLUTION: Expanded data 107 generated by a data expansion part 102 are divided into 64-bit frames (group of blocks) like a 1st section E(sub 1) 108, a 2nd section E(sub 2) 109... and inputted to a single-shot expansion part 105 in order. The single-shot expansion part 105 performs a single-shot expanding process for the 1st section E(sub 1) 108 by using a 256-bit initial value 110 as a parameter while performing conversion and transposition. For the 2nd section E(sub 2) 109, a single-shot expanding process is performed by using a 1st 256-bit intermediate output which is obtained while conversion and transposition are carried out. Consequently, a 2nd 256-bit intermediate output is calculated. This process is repeated until the frame of the final section is inputted and then the 256-bit intermediate output which is calculated finally is used as a hash value Hash 111.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－340048

(43)公開日　平成10年(1998)12月22日

(51)Int.Cl.[6]		識別記号	F I		
Ｇ０９Ｃ	1/00	６５０	Ｇ０９Ｃ	1/00	６５０Ｚ
		６２０			６２０Ｚ
		６４０			６４０Ｄ
Ｈ０４Ｌ	9/06		Ｈ０４Ｌ	9/00	６１１Ｚ
	9/30				６６３Ｚ

審査請求　未請求　請求項の数22　ＯＬ　（全 53 頁）

(21)出願番号　特願平9－149423

(22)出願日　平成９年(1997)６月６日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　宝木　和夫
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　車谷　博之
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内

(74)代理人　弁理士　富田　和子

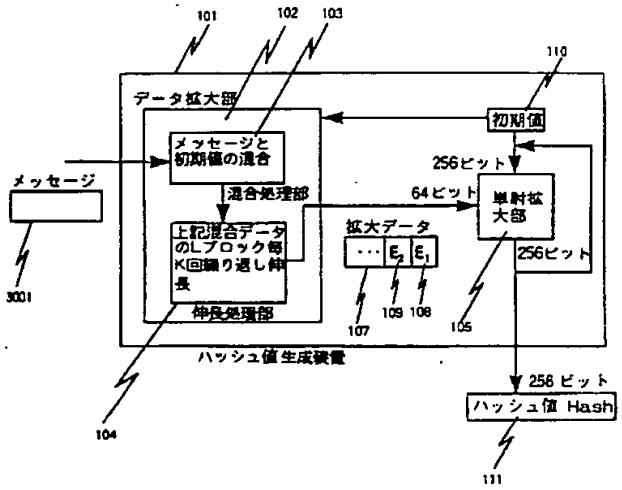
(54)【発明の名称】　ハッシュ値生成方法、データ暗号化方法、データ復号化方法、ハッシュ値生成装置、データ暗号化装置およびデータ復号化装置

(57)【要約】

【課題】データ攪乱度の高いハッシュ値や鍵、あるいは暗号文などを、迅速に生成する。

【解決手段】メッセージが与えられたときに、メッセージの分割データを入力し、それよりも長いデータを出力するような単射拡大処理を行う。また、乗算処理、および巡回シフト処理を行う処理を含むハッシュ関数により、ハッシュ値を生成する。

図１



1

【特許請求の範囲】

【請求項１】電子捺印、あるいはデータ暗号化に用いられるハッシュ値生成方法であって、

対象データを少なくとも２つのブロックに分割する第一の処理と、

前記第一の処理の結果得られた少なくとも２つのブロックのうちのいずれか１つに対して、換字および／または転置変換を行う第二の処理と、

前記第二の処理の結果得られたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う第三の処理と、

前記第三の処理の結果得られたデータを、さらに少なくとも２つのブロックに分割する第四の処理と、

前記第四の処理の結果得られた少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第五の処理と、

を含んでいることを特徴とするハッシュ値生成方法。

【請求項２】電子捺印、あるいはデータ暗号化に用いられるハッシュ値生成方法であって、

対象データを少なくとも２つに分割する第一の処理と、

前記第一の処理の結果得られた少なくとも２つのブロックのうちの、いずれか少なくとも１つに対して、入力値が異なれば出力値も必ず異なり（単射）、かつ出力値の長さが入力値の長さよりも長くなる（拡大）変換である単射拡大変換を行う第二の処理と、

を含んでいることを特徴とするハッシュ値生成方法。

【請求項３】請求項２記載のハッシュ値生成方法であって、

単射拡大変換は、前記第一の処理の結果得られた少なくとも２つのブロックのうち、当該単射拡大変換の対象となるデータ以外のデータを、パラメータとすることを特徴とするハッシュ値生成方法。

【請求項４】請求項２記載のハッシュ値生成方法であって、

単射拡大変換は、前記第一の処理の結果得られた少なくとも２つのブロックとは別個に設定された初期値をパラメータとすることを特徴とするハッシュ値生成方法。

【請求項５】請求項２記載のハッシュ値生成方法であって、

前記第二の処理は、前記第一の処理の結果得られた少なくとも２つのブロックのうちの、いずれか少なくとも２つに対して、単射拡大変換を行うものであり、

当該いずれか少なくとも２つのうちの１つに対して行われる単射拡大変換は、前記第一の処理の結果得られた少なくとも２つのブロックとは別個に設定された初期値をパラメータとするものであり、

他の１つに対して行われる単射拡大変換は、前記初期値をパラメータとする単射拡大変換の結果をパラメータとするものであることを特徴とするハッシュ値生成方法。

【請求項６】請求項２記載のハッシュ値生成方法であって、

2

前記第二の処理は、前記第一の処理の結果得られた少なくとも２つのブロックのうちの、いずれか少なくとも２つに対して、単射拡大変換を行うものであり、かつそのうちの１つに対しては、単射拡大変換を２回行うものであり、

当該そのうちの１つに対して２回行われる単射拡大変換は、他の１つに対して行われる単射拡大変換の結果をパラメータとすることを特徴とするハッシュ値生成方法。

【請求項７】請求項２記載のハッシュ値生成方法であって、

前記単射拡大変換は、入力値に対して換字および／あるいは転置変換を行う第三の処理と、

前記第三の処理の結果得られた入力値に対して、結果が当該データのデータ長よりも長くなるような乗算を行う第四の処理と、

を含んでいることを特徴とするハッシュ値生成方法。

【請求項８】請求項２記載のハッシュ値生成方法であって、

前記単射拡大変換は、入力値に対して換字および／あるいは転置変換を行う第三の処理と、

前記第三の処理の結果得られた入力値に対して、巡回シフト計算を行う第四の処理と、

を含んでいることを特徴とするハッシュ値生成方法。

【請求項９】一定長のデータを暗号変換して一定長の暗号化データを出力するデータ暗号化方法であって、

対象データを換字および／または転置変換を行う第一の処理と、

前記第一の処理の結果得られたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う第二の処理と、

前記第二の処理の結果得られたデータを、少なくとも２つのブロックに分割する第三の処理と、

前記第三の処理の結果得られた少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第四の処理と、

含んでいることを特徴とするデータ暗号化方法。

【請求項１０】データを暗号変換して暗号化データを出力するデータ暗号化方法であって、

対象データの一部を圧縮変換する第一の処理と、前記第一の処理の結果得られたデータに対して、入力値が異なれば出力値も必ず異なる変換である単射変換を行う第二の処理と、前記第二の処理の結果得られたデータを、暗号化データの一部として出力する第三の処理と、を含む暗号化処理を、当該対象データを構成する全ての部分に対して順次行うものであり、

前記第一の処理は、対象データを構成する最初の一部に対しては、鍵を変換して得られるデータをパラメータとして圧縮変換を行い、対象データを構成する二番目以降の一部に対しては、一つ前の暗号化処理において行われ

た第二の処理の結果をパラメータとして圧縮変換を行うものであり、
前記第二の処理は、単射変換の過程において乗算を２回以上行うものであることを特徴とするデータ暗号化方法。
【請求項１１】公開鍵を用いて平文を暗号化する公開鍵暗号を用いたデータ暗号化方法であって、
第一の公開鍵を変換することで得られたデータをパラメータとして、平文に対して暗号化変換を行う第一の処理と、
少なくとも１つの第二の公開鍵に応じたデータと、前記第一の公開鍵を変換することで得られたデータとの間において、前記第二の公開鍵に応じたデータが分かれば、前記第一の公開鍵を変換することで得られたデータを、直接あるいは間接に求めることができる関係式を満たすデータ値を生成する第二の処理と、を含み、
送信すべき暗号化データとして、前記第一の処理の結果得られたデータに、前記第二の処理の結果得られたデータ値を付与したことを特徴とするデータ暗号化方法。
【請求項１２】請求項１１記載のデータ暗号化方法であって、
前記第一の処理は、生成した乱数と第一の公開鍵とを作用させることで得られたデータを、パラメータとして用いるものであることを特徴とするデータ暗号化方法。
【請求項１３】請求項１１記載のデータ暗号方法であって、
前記第一の処理は、複数に区分に分割された平文に対して、区分単位で順次暗号化変換を行うものであり、
かつ、最初の区分に対しては、生成した乱数と第一の公開鍵とを作用させることで得られたデータを、パラメータとして、暗号化変換を行い、
二番目以降の区分に対しては、一つ前の区分に対する暗号化データ生成の過程で発生した中間データを、パラメータとして、暗号化変換を行うものでることを特徴とするデータ暗号化方法。
【請求項１４】秘密鍵を用いて暗号文を復号化する公開鍵暗号を用いたデータ復号化方法であって、
当該方法は、請求項１１記載のデータ暗号化方法によって得られた暗号文を復号化するものであり、
請求項１１記載の第二の公開鍵に応じたデータを、当該第二の公開鍵と対になる秘密鍵から求める第三の処理と、
暗号文に付加されたデータ値と、前記第三の処理で求めたデータとを基に、請求項１１において第一の公開鍵を変換することで得られたデータを求める第四の処理と、
前記第四の処理で求めたデータをパラメータとして、暗号文を復号する第五の処理と、
を含んでいることを特徴とするデータ復号化方法。
【請求項１５】請求項１４記載のデータ復号化方法であって、

当該方法は、請求項１３記載のデータ暗号化方法によって得られた暗号文を復号化するものであり、
前記第五の処理は、複数に区分に分割された暗号文に対して、区分単位で順次復号化変換を行うものであり、
かつ、最初の区分に対しては、前記第四の処理で求めたデータをパラメータとして、復号化変換を行い、
二番目以降の区分に対しては、一つ前の区分に対する復号化データ生成の過程で発生した中間データを、パラメータとして、復号化変換を行うものであることを特徴とするデータ復号化方法。
【請求項１６】公開鍵暗号を用いたデータ暗号化・復号化方法であって、
送信側において、
楕円曲線上のベースポイントＰのｋ倍点ｋＰのｘ座標値と、公開鍵Ｑのｋ倍点ｋＱのｘ座標と、平文とを作用させることで、暗号文を生成するとともに、
当該生成した暗号文に前記ｋＰのｘ座標値を付加して送信し、
受信側において、
受信した暗号文と、当該暗号文に付加されたｋＰのｘ座標値に対応する楕円曲線上の２点のうちのいずれか一方の点Ｒを公開鍵Ｑと対になる秘密鍵ｄでｄ倍した点ｄＲのｘ座標値と、を作用させることで、平文を生成することを特徴とするデータ暗号化・復号化方法。
【請求項１７】元のデータへの逆変換が困難であるハッシュ値を生成するハッシュ値生成装置であって、
対象データを少なくとも２つのブロックに分割する第一の分割手段と、
前記第一の分割手段で分割され少なくとも２つのブロックのうちのいずれか１つに対して、換字および／または転置変換を行う第一の換字・転置変換手段と、
前記第一の換字・転置変換手段で変換されたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う乗算手段と、
前記乗算手段で乗算されたデータを、さらに少なくとも２つのブロックに分割する第二の分割手段と、
前記第第二の分割手段で分割された少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第二の換字・転置変換手段と、
を含んでいることを特徴とするハッシュ値生成装置。
【請求項１８】元のデータへの逆変換が困難であるハッシュ値を生成するハッシュ値生成装置であって、
対象データを少なくとも２つに分割する第一の分割手段と、
前記第一の分割手段で分割された少なくとも２つのブロックのうちの、いずれか少なくとも１つに対して、入力値が異なれば出力値も必ず異なり（単射）、かつ出力値の長さが入力値の長さよりも長くなる（拡大）変換である単射拡大変換を行う単射拡大変換手段と、
を含んでいることを特徴とするハッシュ値生成装置。

5

【請求項１９】一定長のデータを暗号変換して一定長の暗号化データを出力するデータ暗号化装置であって、
対象データに対して換字および／または転置変換を行う第一の換字・転置変換手段と、
前記第一の換字・転置変換手段で変換されたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う乗算手段と、
前記乗算手段で乗算されたデータを、少なくとも２つのブロックに分割する分割手段と、
前記分割手段で分割された少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第二の換字・転置変換手段と、
含んでいることを特徴とするデータ暗号化装置。
【請求項２０】データを暗号変換して暗号化データを出力するデータ暗号化装置であって、
対象データの一部を圧縮変換する圧縮手段と、前記圧縮手段で圧縮されたデータに対して、入力値が異なれば出力値も必ず異なる変換である単射変換を行う単射変換手段と、前記第単射変換手段で変換されたデータを、暗号化データの一部として出力する出力手段と、を含む暗号化手段と、
当該対象データを構成する全ての部分を、前記暗号化手段に順次入力する手段と、を有し、
前記圧縮手段は、対象データを構成する最初の一部に対しては、鍵を変換して得られるデータをパラメータとして圧縮変換を行い、対象データを構成する二番目以降の一部に対しては、一つ前に入力されたデータに対する前記単射変換手段での結果をパラメータとして圧縮変換を行うものであり、
前記単射変換手段は、単射変換の過程において乗算を２回以上行うものであることを特徴とするデータ暗号化装置。
【請求項２１】公開鍵を用いて平文を暗号化する公開鍵暗号を用いたデータ暗号化装置であって、
第一の公開鍵を変換することで得られたデータをパラメータとして、平文に対して暗号化変換を行う暗号化手段と、
少なくとも１つの第二の公開鍵に応じたデータと、前記第一の公開鍵を変換することで得られたデータとの間において、前記第二の公開鍵に応じたデータが分かれば、前記第一の公開鍵を変換することで得られたデータを、直接あるいは間接に求めることができる関係式を満たすデータ値を生成するデータ値生成手段と、を含み、
送信すべき暗号化データとして、前記暗号化手段で暗号化されたデータに、前記データ値生成手段で生成されたデータ値を付与することを特徴とするデータ暗号化装置。
【請求項２２】秘密鍵を用いて暗号文を復号化する公開鍵暗号を用いたデータ復号化装置であって、
当該装置は、請求項２１記載のデータ暗号化装置によって生成された暗号文を復号化するものであり、

6

請求項２１記載の第二の公開鍵に応じたデータを、当該第二の公開鍵と対になる秘密鍵から求める第一のデータ算出手段と、
暗号文に付加されたデータ値と、前記第三の処理で求めたデータとを基に、請求項１１において第一の公開鍵を変換することで得られたデータを求める第二のデータ算出手段と、
前記第二のデータ算出手段で求めたデータをパラメータとして、暗号文を復号化する復号化手段と、
を含んでいることを特徴とするデータ復号化装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、電子捺印やデータ暗号化などのコンピュータネットワークにおけるセキュリティを確保する技術に関し、特に、メッセージを逆変換困難な値であるハッシュ値に変換する方法に関する。
【０００２】
【従来の技術】電子メールなどのネットワークを介してやり取りされるデータの暗号化方式として、公開鍵暗号化方式がある。この公開鍵暗号化方式による処理の流れを概説すると、次のとおりである。
【０００３】　ユーザは、自己宛てのメールを暗号化するための公開鍵を、予めメールの送信者に配布しておく。
【０００４】　メールの送信者は、メールの宛先となるユーザによって予め配布された公開鍵を用いて、メールを暗号化する。そして、暗号化されたメールを、当該メールの宛先へ送信する。
【０００５】　自己が配布した公開鍵で暗号化された電子メールを受け取ったユーザは、自己が所有する秘密鍵（公開鍵とは異なる数値を持つ）を用いて、当該電子メールを復号化する。
【０００６】ところで、この公開鍵暗号化方式は、データの暗号化のみならず、ネットワークを用いた商取引（電子商取引）において、電子的に契約などの正当性を証明する技術である電子捺印（デジタル署名）技術にも応用されている。
【０００７】しかしながら、電子捺印技術において、公開鍵暗号だけを用いて長いメッセージに対する電子捺印を生成しようとすると、多くの処理時間を要する。このため、メッセージを、一旦、短いデータに圧縮してから、この圧縮されたデータに対して電子捺印を生成する方法が提案されている。
【０００８】ここで、圧縮の方法は、通常のデータ圧縮のように、圧縮されたデータから元のメッセージを復元できるように圧縮する必要はない。しかし、ある種の暗号的特性を持つように圧縮する必要がある。このような圧縮を実現する関数として、ハッシュ関数(hash function)が提案されている。

【０００９】商取引文書等のメッセージ、たとえば
文書Ａ：「太郎商会殿　乗用車（カタログNo. 1443）を104万円で購入します。1996年3月10日　吉浦」
が、ハッシュ関数への入力データとなる。入力データの長さはどんなに長くても構わない。

【００１０】ハッシュ関数は、この入力データに対して暗号変換に似た処理を施すことで一定長の短いデータになるように圧縮する。たとえば、
ハッシュ値：283AC9081E83D5B28977
が、ハッシュ関数の出力となる。

【００１１】このハッシュ値は、メッセージダイジェスト、あるいはフィンガープリント（指紋）とも呼ばれており、理想的には一つの入力データ（メッセージ）に対して世の中に実質上一つしか存在しない。この「世の中に実質上一つしか存在しない」ことを保証するため、ハッシュ値の長さは少なくとも128ビット程度は必要といわれている。より詳しくいえば、ハッシュ関数は、次のような特性をもつことが必要である。

【００１２】　一方向性(one－way property)
あるハッシュ関数の出力値が与えられたとする。この出力値と同じ出力値をもたらすような別のメッセージを求めることが、計算量的に困難でなければならない。

【００１３】たとえば、和夫君の誕生日が２月２２日であるとする。和夫君の誕生日と同じ誕生日を持つ別の人を探し出すためには、確立的にいえば、３６５／２＝１８３人程度の人の誕生日を調べればよい。

【００１４】このことは、人をメッセージに置き換え、誕生日をハッシュ値に置き換えても、同様に成り立つ。すなわち、ハッシュ値の長さを１６０ビットとすれば、ハッシュ値の総数は2^{160}通りある。あるメッセージのハッシュ値と同じハッシュ値をもつ別のメッセージを探し出すためには、平均して、$2^{160}/2=2^{159}$個のメッセージに調べる必要がある。これは計算量的に困難である。

【００１５】　衝突回避性(collision free property)
メッセージもハッシュ値も何であるかは問わない。とにかく同じハッシュ値となる二つの異なるメッセージを見つけることが計算量的に困難でなければならない。

【００１６】たとえば、とにかく誰でも良いから同じ誕生日を持つ二人を探したい。この場合、確率的にいえば、$365^{1/2}=24$人程度の人の誕生日を調べればよい。

【００１７】このことは、人をメッセージに置き換え、誕生日をハッシュ値に置き換えても、同様に成り立つ。すなわち、ハッシュ値の長さを１６０ビットとすれば、同じハッシュ値をもつ二つの異なるメッセージ（どんなメッセージでもよい）を探し出すためには、平均して、$2^{160/2}=2^{80}$組程度のメッセージを調べる必要がある。この数は、一方向性の場合と比べるとかなり小さい。しかし、依然、計算量的には困難である。

【００１８】上記のような特性を必要とするハッシュ関数を実現する方法として、種々の方法が発表されているが、現在、換字と転置の繰り返しを行うことでハッシュ値を得る方法が主流になっている。その処理の仕組みを示す従来例が、次の文献１において開示されている。

【００１９】ISO/IEC 10118-2, "Information technology　Security techniques - Hash-functions: Part 2: Hash-functions using an n-bit block cipher algorithm" (1994)

上記文献１に開示されているハッシュ関数について、図面を用いて説明する。

【００２０】図２７は従来のハッシュ関数を説明するための図であり、図２７（ａ）は一般的なハッシュ関数の処理の流れを説明するための図、図２７（ｂ）は図２７（ａ）に示す換字・転置繰り返し処理３００５に、ＤＥＳ（Data Encryption Standard）のような暗号化関数を用いた場合の処理の流れを説明するための図である。

【００２１】図２７（ａ）において、圧縮しようとしているメッセージ３００１は、第１区分$P_1$３００２、第２区分$P_2$３００３、・・・というように、一定の長さ毎に分割され、順次、ハッシュ関数３００７に入力される。

【００２２】ハッシュ関数３００７は、初期値３００４をパラメータとして、第１区分$P_1$３００２に対して換字・転置繰り返し処理３００５を施す。これにより、第１番目の中間出力を算出する。

【００２３】次に、その第１番目の中間出力をパラメータ（初期値３００４の代わり）にして、第２区分$P_2$３００３に対して換字・転置繰り返し処理３００５を施す。これにより、第２番目の中間出力を算出する。

【００２４】上記の処理を最後の区分のデータが入力されるまで繰り返すことで、最後に算出された中間出力を、ハッシュ値Hash３００６として用いる。

【００２５】ここで、上記文献１では、換字・転置繰り返し処理３００５として、米国暗号標準ＤＥＳのような暗号化関数（ブロック暗号）を用いている。このようなハッシュ関数は、「ブロック暗号を利用したハッシュ関数」と呼ばれおり、ＩＳＯ（International Organization for Standardization）で標準化済みである。

【００２６】この「ブロック暗号を利用したハッシュ関数」の詳細は次のとおりである。

【００２７】図２７（ｂ）において、初期値３００４を変換関数３００８で変換したものを、パラメータとして、第１区分$P_1$３００２を暗号化関数３００９に入力する。そして、暗号化関数３００９での暗号化結果と、第１区分$P_1$３００２との間で、ビット毎に排他的論理和３０１０を施す。これにより、換字・転置繰り返し処理３００５での第１番目の中間出力を算出する。

【００２８】次に、第１番目の中間出力をフィードバックして、これを変換関数３００８で変換したものをパラ

メータとして、第２区分$P_2$３００３を暗号化関数３００９に入力する。そして、暗号化関数３００９での暗号化結果と、第２区分$P_2$３００３との間で、ビット毎に排他的論理和３０１０を施す。これにより、換字・転置繰り返し処理３００５での第２番目の中間出力を算出する。

【００２９】上記の処理を最後の区分のデータが入力されるまで繰り返すことで、最後に算出された中間出力を、ハッシュ値Hash３００６として用いる。

【００３０】図２７（ｂ）に示す「ブロック暗号を利用したハッシュ関数」において、暗号化関数３００９としてＤＥＳなどを用いる場合、通常、第１区分$P_1$３００２、第２区分$P_2$３００３、・・・の各区分の長さ、および、換字・転置繰り返し処理３００５の出力の長さは、各々６４ビットとなる。したがって、ハッシュ値Hash３００６の長さも６４ビットとなる。

【００３１】この「ブロック暗号を利用したハッシュ関数」の特徴は、メッセージの各区分$P_1$３００２、$P_2$３００３、・・・の長さと、換字・転置繰り返し処理３００５の出力の長さが等しいことである。

【００３２】なお、換字・転置繰り返し処理３００５として、ＤＥＳのような暗号化関数を用いないハッシュ関数も提案されている。このようなハッシュ関数は、「専用ハッシュ関数」と呼ばれおり、インターネット標準のＭＤ５や、ＩＳＯで標準化中のＳＨＡ－１、ＲＩＰＥＭＤ－１６０などがある。

【００３３】このうち、ＭＤ５については、次の文献２で開示されている。

【００３４】R.Rivest,"The MD5 Message-Digest Algorithm," IETF RFC 1321 (1992)

なお、ＭＤ５の処理の流れ自体は、図２７（ａ）に示すものと同じであるので、この図を用いて説明する。

【００３５】まず、圧縮しようとしているメッセージ３００１は、第１区分$P_1$３００２、第２区分$P_2$３００３、・・・というように、５１２ビット毎に分割され、順次、ハッシュ関数３００７に入力される。

【００３６】ハッシュ関数３００７は、１２８ビットの初期値３００４をパラメータとして、第１区分$P_1$３００２に対して、単純な換字・転置繰り返し処理３００５を施す。これにより、１２８ビットの第１番目の中間出力を算出する。

【００３７】次に、その第１番目の中間出力をパラメータ（初期値３００４の代わり）にして、第２区分$P_2$３００３に対して、単純な換字・転置繰り返し処理３００５を施す。これにより、１２８ビットの第２番目の中間出力を算出する。

【００３８】上記の処理を最後の区分のデータが入力されるまで繰り返すことで、最後に算出された１２８ビットの中間出力を、ハッシュ値Hash３００６として用いる。

【００３９】この「専用ハッシュ関数」の特徴は、メッセージの各区分$P_1$３００２、$P_2$３００３、・・・の長さよりも、換字・転置繰り返し処理３００５の出力の長さが短いことである

【００４０】

【発明が解決しようとする課題】以上説明した従来技術には、以下のような問題点がある。

【００４１】（１）従来提案されているハッシュ関数の問題点

「ブロック暗号を利用したハッシュ関数」の問題点

上述したように、「ブロック暗号を利用したハッシュ関数」では、ＤＥＳなどの暗号化関数（ブロック暗号）を用いている。ブロック暗号は、通常、入力および出力のデータ長が６４ビットである。このため、ハッシュ値も６４ビットとなる。一方、上述したように、一つの入力データ（メッセージ）に対して、ハッシュ値が、「世の中に実質上一つしか存在しない」ことを保証するためには、ハッシュ値の長さを、少なくとも１２８ビット程度以上にする必要があるといわれている。

【００４２】したがって、「ブロック暗号を利用したハッシュ関数」において、１２８ビットのハッシュ値を得ようとする場合、ブロック暗号への入力データ（６４ビット）各々に対して、初期値などを変えて、当該ブロック暗号処理を２回行う必要がある。すなわち、ブロック暗号への入力データ（６４ビット）各々に対して、出力（６４ビット）を２回算出する必要がある。これでは、ハッシュ値生成の処理速度が遅くなるという不満が生じる。

【００４３】　「専用ハッシュ関数」の問題点

「専用ハッシュ関数」では、「ブロック暗号を利用したハッシュ関数」と異なり、メッセージを分割したデータ各々に対して、換字・転置繰り返し処理を２回行わなくても、１２８ビットのハッシュ値を得ることができる。

【００４４】しかしながら、「専用ハッシュ関数」では、上述したように、メッセージを分割したデータ各々に対して、単純な換字・転置繰り返し処理を施すことで、ハッシュ値を得ている。ここで、換字・転置繰り返し処理の出力値の長さ（上記の例では、５１２ビット）は、その入力値の長さ（上記の例では、１２８ビット）より短い。すなわち、換字・転置繰り返し処理において、圧縮が行われていることになる。

【００４５】このため、たとえば、５１２ビット毎に複数のデータに分割した場合に、最後の区分のデータのみが異なるような２つのメッセージを想定した場合、単純な換字・転置繰り返し処理により最後の区分のデータ（５１２ビット）を１２８ビットの出力に圧縮する過程において、その出力（すなわちハッシュ値）が同じになる確立が高まる。これでは、衝突回避性を劣化させてしまう。

【００４６】　なお、上記　、　で述べた問題点は、ハ

ッシュ関数を電子捺印に用いた場合にのみ生じる問題ではない。たとえば、データの暗号化方式に用いた場合でも、同様に生じる問題である。

【００４７】（２）公開鍵暗号方式の問題点

上述したように、公開鍵暗号を用いて長いデータを暗号化しようとすると、多くの処理時間を要する。

【００４８】公開鍵暗号方式を電子メールなどのデータ暗号化に適用した場合、同じ電子メールを複数の宛先に暗号化して送信しようとすると、送信者は、各宛先毎に、当該宛先から予め配布されて公開鍵を用いて暗号化処理を行わなければならない。すなわち、電子メールの暗号化処理を宛先の数だけ繰り返し行わなければならない。

【００４９】一方、受信者は、間違ってファイルから復号鍵を消去するなどして、復号鍵を紛失してしまった場合、自己宛てに送られてきた、自己が配布した公開鍵で暗号化された電子メールを復号することができなくなってしまう。

【００５０】本発明は上記事情に鑑みてなされたものであり、本発明の目的はデータ撹乱度の高いハッシュ値や鍵、あるいは暗号文などを、迅速に生成する方法および装置を提供することにある。

【００５１】また、本発明の他の目的は、間違ってファイルから復号鍵を消去するなどして、復号鍵を紛失してしまった場合でも、自己宛てに送られてきた暗号化データを、他の二人以上の受信者が協力することで復号することが可能なデータ暗号化方法および装置を提供することにある。

【００５２】

【課題を解決するための手段】上記課題を解決するために、本発明のハッシュ値生成方法および装置は、対象データを少なくとも２つのブロックに分割する第一の構成要素と、前記第一の構成要素の結果得られた少なくとも２つのブロックのうちのいずれか１つに対して、換字および／または転置変換を行う第二の構成要素と、前記第二の構成要素の結果得られたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う第三の構成要素と、前記第三の構成要素の結果得られたデータを、さらに少なくとも２つのブロックに分割する第四の構成要素と、前記第四の構成要素の結果得られた少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第五の構成要素と、を含む。

【００５３】本発明のハッシュ値生成方法および装置では、上述したように、ハッシュ値生成過程において、入力値に対して出力値の長さが長くなるような乗算を行っている。乗算処理によれば、出力値のビット各々が入力値のビット各々の影響を受けるので、データの撹乱を効率よく行うことができる。

【００５４】ところで、乗算処理は、最近のマイクロプロセッサの進展により取り分け処理速度が向上している。したがって、データ撹乱度の高いハッシュ値の生成を迅速に行うことができる。

【００５５】また、本発明の他のハッシュ値生成方法および装置は、対象データを少なくとも２つに分割する第一の構成要素と、前記第一の構成要素の結果得られた少なくとも２つのブロックのうちの、いずれか少なくとも１つに対して、入力値が異なれば出力値も必ず異なり（単射）、かつ出力値の長さが入力値の長さよりも長くなる（拡大）変換である単射拡大変換を行う第二の構成要素と、を含む。

【００５６】本発明の他のハッシュ値生成方法および装置では、ハッシュ値生成の過程において、出力の長さを入力の長さより長くし、かつ、入力値が異なれば出力値も必ず異なるような単射拡大変換を行っている。このため、衝突回避性の高い、すなわち安全なハッシュ値を生成することができる。

【００５７】ここで、対象データは、単射拡大変換でのパラメータとして用いる初期値と混合させてから、第一の構成要素に入力するようにしてもよい。このようにすることで、異なった初期値に対して同じハッシュ値が導かれるといったような初期値の衝突が、起こる確率を低くすることができる。

【００５８】また、第一の構成要素に入力した対象データを、再度、第一の構成要素に入力するようにしてもよい。このようにすることで、異なったメッセージに対して同じハッシュ値が導かれるといったようなメッセージの衝突が、起こる確率を低くすることができる。

【００５９】また、本発明のデータ暗号化方法および装置は、一定長のデータを暗号変換して一定長の暗号化データを出力するものであり、対象データを換字および／または転置変換を行う第一の構成要素と、前記第一の構成要素の結果得られたデータに対して、結果が当該データのデータ長よりも長くなるような乗算を行う第二の構成要素と、前記第二の構成要素の結果得られたデータを、少なくとも２つのブロックに分割する第三の構成要素と、前記第三の構成要素の結果得られた少なくとも２つのブロック各々に対して、換字および／または転置変換を行う第四の構成要素と、含む。

【００６０】この構成においても、データ暗号化の過程において、入力値に対して出力値の長さが長くなるような乗算を行っているので、データの撹乱を効率よく行うことができる。

【００６１】さらに、本発明の他のデータ暗号化方法および装置は、公開鍵を用いて平文を暗号化する公開鍵暗号を用いたものであり、第一の公開鍵を変換することで得られたデータをパラメータとして、平文に対して暗号化変換を行う第一の構成要素と、少なくとも１つの第二の公開鍵に応じたデータと、前記第一の公開鍵を変換することで得られたデータとの間において、前記第二の公開鍵に応じたデータが分かれば、前記第一の公開鍵を変

換することで得られたデータを、直接あるいは間接に求めることができる関係式を満たすデータ値を生成する第二の構成要素と、を含み送信すべき暗号化データとして、前記第一の処理の結果得られたデータに、前記第二の処理の結果得られたデータ値を付与したことを特徴とする。

【００６２】また、このデータ暗号化方法および装置と対になるデータ復号化方法および装置は、第二の公開鍵に応じたデータを、当該第二の公開鍵と対になる秘密鍵から求める第三の構成要素と、暗号文に付加されたデータ値と、前記第三の構成要素で求めたデータとを基に、第一の公開鍵を変換することで得られたデータを求める第四の構成要素と、前記第四の構成要素で求めたデータをパラメータとして、暗号文を復号する第五の構成要素と、を含む。

【００６３】この構成によれば、第二の公開鍵と対になる秘密鍵を有する者は、単独あるいは他の第二の公開鍵を有する者と協力することで、第二の構成要素で生成されたデータ値から、第一の公開鍵を変換することで得られたデータを得ることができる。

【００６４】したがって、第一の公開鍵と対の秘密鍵を有する者は勿論のこと、第二の公開鍵と対になる秘密鍵を有する者であっても、データの復号が可能となる。

【００６５】このことは、送信側にとっては、同じ電子メールを複数の宛先に暗号化して送信しようとする場合に、各宛先毎に、当該宛先各々から予め配布された公開鍵を用い、電子メールの暗号化処理を一々行なう必要がなくなることを意味する。

【００６６】

【発明の実施の形態】以下に、本発明の第一実施形態について説明する。

【００６７】図１は本発明の第一実施形態であるハッシュ値生成装置の機能構成を示す図である。このハッシュ値生成装置は、たとえば、パーソナルコンピュータやＩＣカードなどのマイクロプロセッサを備えた情報処理装置において、マイクロプロセッサに所定のプログラムを実行させることで実現可能である。また、１つのＬＳＩで実現させることも可能である。なお、このハッシュ値生成装置は、電子捺印や電子メールなどのデータ暗号化に用いることができる。

【００６８】図１において、圧縮しようとしているメッセージ３００１が、ハッシュ値生成装置１０１に入力されると、まず、データ拡大部１０２において、以下の処理が行われる。

【００６９】　混合処理部１０３において、入力されたメッセージ３００１と初期値との混合処理が行われる。この混合処理については後述する。

【００７０】　伸長処理部１０４において、混合処理部１０３で得た混合データのＬブロック毎Ｋ回繰り返し伸長処理が行われる。このＬブロック毎Ｋ回繰り返し伸長処理については後述する。

【００７１】上記　、　の処理によりメッセージ３００１の拡大データ１０７を生成する。

【００７２】データ拡大部１０２で生成された拡大データ１０７は、第１区分$E_1$１０８、第２区分$E_2$１０９、・・・というように、６４ビット毎のフレーム（ブロックのかたまり）に分割されて、順次、単射拡大部１０５に入力される。

【００７３】単射拡大部１０５は、２５６ビットの初期値１１０をパラメータとして、第１区分$E_1$１０８に対して、換字・転置を行いながら単射拡大処理（これについては後述する）を施す。これにより、２５６ビットの第１番目の中間出力を算出する。

【００７４】次に、その第１番目の中間出力をパラメータ（初期値１１０の代わり）にして、第２区分$E_2$１０９に対して、換字・転置を行いながら単射拡大処理を施す。これにより、２５６ビットの第２番目の中間出力を算出する。

【００７５】上記の処理を最後の区分のフレームが入力されるまで繰り返すことで、最後に算出された２５６ビットの中間出力を、ハッシュ値Hash１１１として用いる。

【００７６】次に、図１に示すハッシュ値生成装置１０１の各部での処理について、詳述する。

【００７７】まず、データ拡大部１０２の混合処理部１０３での処理について説明する。

【００７８】混合処理部１０３は、メッセージ３００１および初期値１１０を各々複数のデータブロックに分割して、両者を混ぜ合わせるといった処理を行う。

【００７９】図２は、混合処理部１０３での処理の一例を説明するための図である。

【００８０】ここで、初期値１１０は、４個の６４ビットデータブロック$I_1$２０１、$I_2$２０２、$I_3$２０３、$I_4$２０４がこの順番に連なって構成されたものとする。

【００８１】まず、パディング処理部２２０において、メッセージ３００１の長さに初期値１１０の長さ（２５６ビット）を加えた値が、Ｌ×６４の整数倍となるように、メッセージ３００１を加工する。

【００８２】ここで、Ｌとは、後述する伸長処理部１０４において行われるＬブロック毎Ｋ回繰り返し伸長処理によって定義される値である。また、６４とは、単射拡大部１０５に入力されるフレーム（１つの区分のデータ）のビット長である。

【００８３】パディング処理部２２０は、具体的には、以下のようにして、メッセージ３００１を加工する。

【００８４】　メッセージ３００１の長さに初期値１１０の長さ（２５６ビット）を加えた値が、Ｌ×６４の整数倍である場合、メッセージ３００１の後部に、ビット「１１」とＬ×６４－２個のビット「０１０１・・・」

とを、この順番で接続する。

【００８５】　メッセージ３００１の長さに初期値１１０の長さ（２５６ビット）を加えた値が、Ｌ×６４ビットの整数倍でない場合、メッセージ３００１の後部に、ビット「１１」と、０個～Ｌ×６４－１個の間でいくつかのビット「０１０１」とを、この順番に接続する。このようにすることで、メッセージ３００１全体の長さが、Ｌ×６４の整数倍となるようにする。

【００８６】次に、パディング処理部２２０にて、全体の長さがＬ×６４の整数倍となるように加工されたメッセージ３００１は、Ｎ個の６４ビットデータブロック$M_1$２０５、$M_2$２０６、$M_3$２０７、$M_4$２０８、$M_5$２０９、・・・がこの順番に連なったデータ２１６に変換される。

【００８７】その後、処理部２１７において、このＮ個の６４ビットデータブロックと、初期値１１０を構成する４個の６４ビットデータブロックとを混ぜ合わせる。具体的には、図２に示すように、

「M_1→D_1、I_1→D_2、M_2→D_3、I_2→D_4、M_3→D_5、I_3→D_6、M_4→D_7、I_4→D_8、M_5→D_9、M_6→D_{10}、M_7→D_{11}、・・・・・」

といったデータの置き換えが行われる。

【００８８】この置き換えた結果としてのＮ＋４個の６４ビットデータブロック$D_1$２１０、$D_2$３１１、$D_3$２１２、$D_4$２１３、$D_5$２１４、・・・がこの順番で連なったものが、中間拡大データ２１５として出力される。

【００８９】この中間拡大データ２１５の長さは、Ｌ×６４ビットの整数倍となる。

【００９０】次に、データ拡大部１０２の伸長処理部１０４での処理について説明する。

【００９１】伸長処理部１０４は、中間拡大データ２１５の伸長処理を行う。

【００９２】図３は、伸長処理部１０４での処理の一例を説明するためのフロー図である。

【００９３】ここでは、中間拡大データ２１５のＬブロック分をコピーし、これをコピーしたブロックのうちの最後のブロックの後に追加するといった動作を、Ｌブロック毎にＫ回繰り返して行っている。本実施形態では、この処理をＬブロック毎Ｋ回繰り返し伸長処理と称している。

【００９４】まず、ステップ３０２において、混合処理部１０３で求めた中間拡大データ２１５を構成するＮ＋４個の６４ビットデータブロック$D_1$２１０、$D_2$３１１、$D_3$２１２、$D_4$２１３、$D_5$２１４、・・・を入力する。

【００９５】次に、ステップ３０３において、ｉ＝１、ｊ＝０を設定する。

【００９６】次に、ステップ３０４において、次式によりｍを求める。

【００９７】ｍ＝（ｉ－（ｉ（mod Ｌ・Ｋ））／Ｋ＋（（ｉ－１）（mod Ｌ））＋１

ここで、mod Ｘとは、Ｘで割った余りをとるという処理を示す。たとえば、５（mod ２）＝１である。

【００９８】次に、ステップ３０５において、中間拡大データ２１５を構成するＮ＋４個の６４ビットデータブロックのうち、ｍ番目のデータブロックD_mを、上述した拡大データ１０７を構成するフレームのうちのｉ番目のフレームE_iに設定する。

【００９９】次に、ステップ３０６において、ステップ３０５で設定したフレームE_iを出力する。

【０１００】次に、ステップ３０７において、ｍ番目のデータブロックD_mが、中間拡大データ２１５を構成するＮ＋４個の６４ビットデータブロックのうちの最後のブロックに相当するか否かを判断する。最後のブロックである場合は、ステップ３０８へ移行し、そうでない場合は、ステップ３１０へ移行する。

【０１０１】ステップ３０８では、ｊの値を１つインクリメント（ｊ＝ｊ＋１）し、ステップ３０９へ移行する。

【０１０２】ステップ３０９では、ｊがＫより大きいか否かを判断する（ｊ＞Ｋ？）。ｊがＫより大きい場合はこのフローを終了し、ｊがＫ以下の場合はステップ３１０へ移行する。

【０１０３】ステップ３１０では、ｉの値を１つインクリメント（ｉ＝ｉ＋１）し、その後、ステップ３０４へ戻る。

【０１０４】上記のフローを実行することで、中間拡大データ２１５に対して、上記説明したＬブロック毎Ｋ回繰り返し伸長処理が施され、結果として、図１に示す拡大データ１０７を構成するフレーム$E_1$１０８、$E_2$１０９、・・・が順次出力される。なお、拡大データ１０７は、中間拡大データ２１５のＫ倍になる。

【０１０５】次に、単射拡大部１０５での処理について説明する。

【０１０６】単射拡大部１０５は、当該単射拡大部１０５に入力される拡大データ１０７に対して換字・転置処理を行うという点では、従来の「ブロック暗号を利用したハッシュ関数」や「専用ハッシュ関数」と同じである。

【０１０７】しかしながら、本実施形態の単射拡大部１０５は、当該単射拡大部１０５に入力された、拡大データ１０７を構成するフレーム各々について、入力値が異なれば出力値も必ず異なり（単射）、且つ入力値の長さよりも出力値の長さの方が長くなる（拡大）ように、入力されたフレームを変換する処理を行う点で、従来のハッシュ関数と異なる。なお、本実施形態では、この処理を単射拡大処理と称している。

【０１０８】図４は、単射拡大部１０５での処理の一例を説明するためのフロー図である。

【０１０９】まず、ステップ４０２において、２５６ビ

ットの初期値１１０を入力する。そして、これをHに設定する。

【０１１０】次に、ステップ４０３において、ｑ＝１に設定する。

【０１１１】次に、ステップ４０４において、図１に示す拡大データ１０７のうち、ｑ番目のフレームE_qを入力する。

【０１１２】次に、ステップ４０５において、フレームE_qに対し、Hをパラメータとして、その長さを６４ビットから９６ビットへ拡大するような単射拡大処理を施す。

【０１１３】次に、ステップ４０６において、ステップ４０５で得たデータに対し、Hをパラメータとして、その長さを９６ビットから１２８ビットへ拡大するような単射拡大処理を施す。

【０１１４】次に、ステップ４０７において、ステップ４０６で得たデータに対し、Hをパラメータとして、その長さを１２８ビットから２５６ビットへ拡大するような単射拡大処理を施す。そして、得た２５６ビットのデータをHに設定する。

【０１１５】次に、ステップ４０８では、ｑ番目のフレームE_qが、拡大データ１０７を構成するフレームE_1、E_2・・・のうちの最後のフレームに相当するか否かを判断する。最後のフレームである場合は、ステップ４１０へ移行し、そうでない場合は、ステップ４０９へ移行する。

【０１１６】ステップ４０９では、ｑの値を１つインクリメント（ｑ＝ｑ＋１）し、その後、ステップ４０４へ戻る。

【０１１７】ステップ４１０では、ステップ４０７で設定した２５６ビットのデータHが、拡大データ１０７を構成するフレームのうちの最後のフレームに対するものであるので、このHをハッシュ値Hash１１１として出力する。

【０１１８】次に、図４に示すステップ４０５～４０７での単射拡大処理について説明する。

【０１１９】まず、図４に示すステップ４０５での単射拡大処理（６４ビット→９６ビットの単射拡大）について説明する。

【０１２０】図５は、図４に示すステップ４０５における６４ビットデータから９６ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【０１２１】まず、ステップ５０２において、図４に示すステップ４０４で入力された６４ビットのフレームE_qを、上位３２ビットのデータX_1と下位３２ビットのデータY_1とに分割する。

【０１２２】また、２５６ビットのデータH（図４において、ステップ４０４で入力されたフレームが１番目のフレームE_1である場合は、ステップ４０２で設定した値、２番目以降のフレームE_2・・・である場合は、直前に実施されたステップ４０７で設定した値）を、先頭から３２ビット毎に分割して、８個の３２ビットデータH_1、H_2、H_3、・・・H_8を得る。

【０１２３】次に、ステップ５０３において、次式で示される処理を行うことで、X_2、Y_2を生成する。

【０１２４】$X_2=X_1+(Y_1+H_1)^2$

$Y_2=Y_1$

この結果、X_2は６４ビット、Y_2は３２ビットのデータとなる。なお、図５のステップ５０３において、太線の矢印は６４ビットデータの流れを、そして、細線の矢印は、３２ビットデータの流れを示している。

【０１２５】次に、ステップ５０４において、X_2、Y_2を出力し、その後、このフローを終了する。

【０１２６】上記のフローにより、３２ビットのデータX_1および３２ビットのデータX_1からなるフレームE_qを、６４ビットのデータX_2および３２ビットのデータY_2の合計９６ビットのデータに拡大することができ、且つ、X_2、Y_2が与えられたとき、$X_1=X_2-(Y_2+H_1)^2$、および、$Y_1=Y_2$から、X_1、Y_1が一意的に定まる関係、すなわち単射とすることができる。

【０１２７】したがって、上記のフローにより、６４ビットデータから９６ビットデータへの単射拡大を行うことができる。

【０１２８】ただし、図４に示すステップ４０５での処理は、図５に示すものに限定されるものではなく、６４ビットデータから９６ビットデータへの単射拡大処理を行うものであればよい。

【０１２９】次に、図４に示すステップ４０６での単射拡大処理（９６ビット→１２８ビットの単射拡大）について説明する。

【０１３０】図６は、図４に示すステップ４０６における９６ビットデータから１２８ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【０１３１】まず、ステップ６０２において、図５に示すフローで生成した６４ビットデータX_2および３２ビットデータY_2を入力する。また、図５に示すステップ５０２により２５６ビットのデータHを分割することで生成した、８個の３２ビットデータH_1、H_2、H_3、・・・H_8のうち、H_2、H_3、H_4、H_5、H_6、H_7、H_8を入力する。

【０１３２】次に、ステップ６０３において、６４ビットデータX_2を、上位３２ビットのデータX_Hと下位３２ビットのデータX_Lとに分割する。

【０１３３】次に、ステップ６０４において、次式で示される処理を順番に行うことでX_3、Y_3を生成する。

【０１３４】$A=X_L$ eor H_2

$B=X_H+H_3+1$

$C=A\cdot B$

$C'=(C$ eor $Y_2)+(H_4\|H_5)+1$

$C'_H\|C'_L=C'$

$D=rot_5(C'_L)$ eor H_6
$E=rot_{12}(C'_H)-H_7+1$
$F=(D \| E)$
$G=X_2+F+1$
$X_3=C'+G+(H_8 \| H_1)$
$Y_3=G$
ここで、eorは、ビット毎の排他的論理和を示す。たとえば、１１００１０eor ０１１００１＝１０１０１１である。また、＋は加算を示す。ただし、最上位ビットの計算において桁上げが生じた場合は、桁上げ部分を無視する。たとえば、１０１１０１＋１００１００＝０１０００１となる。

【０１３５】また、‖はデータの結合を示す。たとえば、１１１１１１‖０００００＝１１１１１０００００である。また、$rot_T(U)$は、数値データUを上位側へTビット巡回シフトすることで得られるデータを示す。たとえば、rot_2（１１００００）＝００００１１となる。ここで、数値データの左側が上位側である。

【０１３６】上記の式で示される処理を順次行うことにより、X_3およびY_3は、各々６４ビットのデータとなる。なお、図６のステップ６０４において、太線の矢印は６４ビットデータの流れを、そして、細線の矢印は、３２ビットデータの流れを示している。

【０１３７】次に、ステップ６０５において、X_3、Y_3を出力し、その後、このフローを終了する。

【０１３８】上記のフローにより、６４ビットのデータX_2および３２ビットのデータY_2の合計９６ビットのデータを、６４ビットのデータX_3および６４ビットのデータY_3の合計１２８ビットのデータに拡大することができ、且つ、X_3、Y_3が与えられたとき、X_2、Y_2が一意的に定まる関係、すなわち単射とすることができる。

【０１３９】したがって、上記のフローにより、９６ビットデータから１２８ビットデータへの単射拡大を行うことができる。

【０１４０】ただし、図４に示すステップ４０６での処理は、図６に示すものに限定されるものではなく、９６ビットデータから１２８ビットデータへの単射拡大処理を行うものであればよい。

【０１４１】次に、図４に示すステップ４０７での単射拡大処理（１２８ビット→２５６ビットの単射拡大）について説明する。

【０１４２】図７は、図４に示すステップ４０７における１２８ビットデータから２５６ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【０１４３】まず、ステップ７０２において、図６に示すフローで生成した６４ビットデータX_3、Y_3を入力する。また、図５に示すステップ５０２により２５６ビットのデータHを分割することで生成した８個の３２ビットデータH_1、H_2、H_3、H_4、H_5、H_6、H_7、H_8を入力する。

【０１４４】次に、ステップ７０３において、６４ビットデータX_3を、上位３２ビットのデータX_Hと下位３２ビットのデータX_Lとに分割する。

【０１４５】次に、ステップ７０４において、６４ビットデータY_3を、上位３２ビットのデータY_Hと下位３２ビットのデータY_Lとに分割する。

【０１４６】次に、ステップ７０５において、次式で示される処理を順番に行うことで、換字・転置を繰り返しながら、K_1、K_2、・・・K_8を生成する。

【０１４７】$K_1 \| K_3=(H_8 \| H_6)+(X_H \| X_L)+((X_H \text{ eor } Y_H) \| (H_L \text{ eor } Y_L))$
$K_2 \| K_4=(H_7 \| H_5) \text{ eor } ((H_8 \| H_6)+(X_H \| X_L))$
$K_5 \| K_7=(H_4 \| H_2)+(Y_H \| Y_L)+((X_L \text{ eor } Y_H) \| (X_H \text{ eor } Y_L))$
$K_6 \| K_8=(H_3 \| H_1) \text{ eor } ((H_4 \| H_2)+(Y_H \| Y_L))$
この結果、K_1、K_2、K_3、K_4、K_5、K_6、K_7、K_8は、各々３２ビットのデータとなる。なお、図７のステップ７０５において、細線の矢印は、３２ビットデータの流れを示している。

【０１４８】次に、ステップ７０６において、K_1、K_2、K_3、K_4、K_5、K_6、K_7、K_8をこの順序で結合して、Hを生成する（$K_1 \| K_2 \| K_3 \| K_4 \| K_5 \| K_6 \| K_7 \| K_8 \rightarrow H$）。これにより、Hは３２ビット×８＝２５６ビットのデータとなる。

【０１４９】ステップ７０８では、ステップ７０６で生成したHを出力し、その後、このフローを終了する。

【０１５０】上記のフローにより、６４ビットのデータX_3およびY_3の合計１２８ビットのデータを、２５６ビットのデータHに拡大することができ、且つ、Hが与えられたとき、X_3、Y_3が一意的に定まる関係、すなわち単射とすることができる。

【０１５１】したがって、上記のフローにより、１２８ビットデータから２５６ビットデータへの単射拡大を行うことができる。

【０１５２】ただし、図４に示すステップ４０７での処理は、図７に示すものに限定されるものではなく、１２８ビットデータから２５６ビットデータへの単射拡大処理を行うものであればよい。

【０１５３】上記の第一実施形態では、入力されたフレームE_qと、初期値１１０あるいは単射拡大部１０５の出力であるHとを用いて、ハッシュ値を生成する過程において、換字・転置処理を行う単射拡大部１０５において、３２ビットデータ同士の乗算処理を行っている（図５に示すステップ５０３での処理$X_2=X_1+(Y_1+H_1)^2$、および図６に示すステップ６０４での処理C＝A・B）。併せて、３２ビットデータに対する巡回シフト計算をも行っている（図６に示すステップ６０４での処理$D=rot_5(C'_L)$ eor H_6、$E=rot_{12}(C'_H)$

＋H_7＋１）。

【０１５４】このようにすることで、従来の「ブロック暗号を利用したハッシュ関数」や「専用ハッシュ関数」に比べて、データ攪乱度の高いハッシュ値を、迅速に生成することができる。

【０１５５】すなわち、３２ビットデータ同士の乗算（３２ビット×３２ビット→６４ビット）は、出力される６４ビットデータの各ビットが、すべての入力ビットの影響を受ける。このため、データの攪乱度が高いので、効率よく換字処理を行うことができる。

【０１５６】なお、現在、主にパーソナルコンピュータ用のマイクロプロセッサとして普及しているインテル社製の１００ＭＨｚのペンティアムプロセッサならば、１秒間に１千万回の積演算（乗算）を行うことができる。これは、１９８０年代中期に発表されたモトローラ社製の２０ＭＨｚの６８０２０プロセッサでは１秒間に５０万回程度しか積演算を行うことができなかったのに対して、およそ２０倍の高速化となっている。

【０１５７】また、３２ビットデータの巡回シフト計算も、効率よく換字処理を行うという点で有効である。

【０１５８】マイクロプロセッサによる演算処理では、一回の処理で巡回シフト計算、すなわち３２ビットデータの転置処理を実現することができるが、最近のマイクロプロセッサ、たとえば、インテル社製のペンティアムプロセッサによれば、１サイクルでこの巡回シフトを完了する。インテル社製の１００ＭＨｚのペンティアムプロセッサならば、１秒間に１億回の巡回シフト計算処理を行うことができる。これは、１９８０年代中期に発表されたモトローラ社製の２０ＭＨｚの６８０２０プロセッサでは１秒間に２５０万回程度しか巡回シフトを行うことができなかったのに対して、およそ４０倍の高速化となっている。

【０１５９】このように、本発明の第一実施形態は、換字・転置処理を行うに際して、最近の技術革新により特に有利になっているマイクロプロセッサの基本演算を使用することで、データ攪乱度の高いハッシュ値の生成を迅速に行うようにしている。

【０１６０】ところで、従来の「ブロック暗号を利用したハッシュ関数」や「専用ハッシュ関数」では、通常、３２ビットデータ同士の加算を行うことで、換字処理を実現している。この３２ビットデータ同士の加算処理は、インテル社製の１００ＭＨｚのペンティアムプロセッサならば、１秒間に１億回実行することができる。これは、１９８０年代中期に発表されたモトローラ社製の２０ＭＨｚの６８０２０プロセッサでは１秒間に１千万回程度しか加算を行うことができなかったのに対して、およそ１０倍程度しか高速化されていない。

【０１６１】乗算処理は、３２回分の加算処理と、３２回分の巡回シフト処理とを行ったのと同じデータ撹乱効果がある。このことを考慮すれば、インテル社製のペンティアムプロセッサが主流となっている現在においては、加算処理よりも乗算処理を用いる有利性が、さらに増したといえる。

【０１６２】また、本発明の第一実施形態では、入力されたフレームE_qと、初期値１１０あるいは単射拡大部１０５の出力であるＨとを用いて、ハッシュ値を生成する過程において、換字・転置処理を行う単射拡大部１０５において、当該単射拡大部１０５に入力された、拡大データ１０７を構成するフレーム各々について、入力値が異なれば出力値も必ず異なり（単射）、且つ入力値の長さよりも出力値の長さの方が長くなる（拡大）ように、入力されたフレームを変換する処理を行っている。

【０１６３】これにより、衝突回避性の高い、すなわち安全なハッシュ値を生成することができる。

【０１６４】すなわち、従来のハッシュ関数では、換字・転置繰り返し処理へ入力されるメッセージの区分（フレーム）の長さと、出力される中間出力の長さとを比べた場合、中間出力の長さは、入力されるフレームの長さと等しいか（「ブロック暗号を利用したハッシュ関数」）、あるいは短かった（「専用ハッシュ関数」）。

【０１６５】これに対し、本発明の第一実施形態では、上記の単射拡大処理により、中間出力の長さ（２５６ビット）が、入力されるフレームの長さ（６４ビット）より長くなる。したがって、ＭＤ５で問題となったような、換字・転置の繰り返し処理におけるメッセージの衝突回避を比較的容易に実現できる。

【０１６６】また、本発明の第一実施形態では、図１に示すように、初期値１１０を、単射拡大部１０５へ入力する最初のパラメータとしてのみならず、メッセージ３００１を単射拡大部１０５へ入力する前段階の処理として、当該メッセージ３００１を拡大して拡大データ１０７を生成するのにも用いている。

【０１６７】このようにすることで、図２７に示すように、初期値３００４を換字・転置繰り返し処理３００５へ入力する最初のパラメータとしてのみ用いていた従来のハッシュ関数に比べ、異なった初期値に対して同じハッシュ値が導かれるといったような、初期値の衝突が起こる確率を、低減させることができる。

【０１６８】さらに、本発明の第一実施形態では、図１に示すように、メッセージ３００１を単射拡大部１０５へ入力する前段階の処理として、メッセージを分割して複数のブロックを生成し、この生成した複数のブロックの一部あるいは全部をコピーして、元の複数のブロックに混ぜるといった処理（本実施形態では、Ｌブロック毎Ｋ回繰り返し伸長処理と称している）を行っている。

【０１６９】このようにすることで、図２７に示すように、メッセージ３００１を換字・転置繰り返し処理３００５へ入力する前段階の処理として、当該メッセージ３００１を分割して、複数の区分P_1、P_2・・・を生成するのみであった従来のハッシュ関数に比べ、異なったメ

ッセージに対して同じハッシュ値が導かれるといったような、メッセージの衝突が起こる確率を、低減させることができる。

【０１７０】なお、上記の第一実施形態では、換字・転置を行う単射拡大部２５６での３２ビットデータ乗算処理として、図５に示すステップ５０３の$X_2=X_1+(Y_1+H_1)^2$で示される処理と、図６に示すステップ６０４の$C=A\cdot B$で示される処理と、を行うものについて説明した。しかしながら、本発明でいう乗算処理がこれ等の式に限定されるものでないことは、当然のことである。

【０１７１】同様に、本発明でいう巡回シフト計算処理も、図６に示すステップ６０４の$D=rot_5(C'_L)\ eor\ H_6$、$E=rot_{12}(C'_H)+H_7+1$で示される処理に限定されるものではない。

【０１７２】また、上記の実施形態では、単射拡大部１０５へ入力される拡大データ１０７のフレームＥ１、Ｅ２、・・・を６４ビットとし、単射拡大部１０５から出力される中間拡大データを２５６ビット（したがって、ハッシュ値は２５６ビット）としたものについて説明した。しかしながら、本発明はこれに限定されるものではない。

【０１７３】以下に、本発明の第一実施形態の変形例として、単射拡大部へ入力されるフレームを６４ビットとし、単射拡大部からの中間出力を８０ビット（したがって、ハッシュ値は８０ビット）としたものについて説明する。

【０１７４】図８は本発明の第一実施形態の変形例であるハッシュ値生成装置の機能構成を示す図である。ここで、図１に示すハッシュ値生成装置１０１と同じ機能を有するものには、同じ符号を付している。

【０１７５】図８に示すハッシュ値生成装置１０１ａが図１に示すハッシュ値生成装置１０１と異なる点は、２５６ビットの初期値１１０に代えて８０ビットの初期値８０２を用いた点、混合処理部１０３に代えて混合処理部８０１を用いた点、および、単射拡大部１０５に代えて単射拡大部８０３を用いた点である。その他の構成は、図１に示すものと同様である。

【０１７６】図８に示すハッシュ値生成装置１０１ａでは、８０ビットのハッシュ値Hash８０４を生成する。

【０１７７】混合処理部８０１は、メッセージ２５０１および初期値８０２各々を複数のデータブロックに分割して、両者を混ぜ合わせるといった処理を行う点で、図１に示す混合処理部１０３と同様である。ただし、８０ビットの初期値８０２を用いているため、具体的な処理が異なってくる。

【０１７８】図９は、混合処理部８０１での処理の一例を説明するための図である。

【０１７９】ここで、初期値８０２は、６４ビットのデータブロック$I_1$９０１と、１６ビットのデータブロック$I_2$９０２とが、この順番で連なって構成されたものとする。

【０１８０】まず、パディング処理部２２０において、メッセージ２５０１の長さに初期値８０２の長さ（８０ビット）を加えた値が、Ｌ×６４の整数倍となるように、メッセージ２５０１を加工する。この処理は、図２に示す混合処理部１０３でのパディング処理２２０と同様である。

【０１８１】次に、パディング処理部２２０にて、全体の長さがＬ×６４の整数倍となるように加工されたメッセージ２５０１は、Ｎ個の６４ビットデータブロック$M_1$２０５、$M_2$２０６、$M_3$２０７、$M_4$２０８、$M_5$２０９、・・・がこの順番に連なったデータ２１６に変換される。

【０１８２】その後、処理部９０３において、このＮ個の６４ビットデータブロックと、初期値８０２を構成する６４ビットのデータブロック$I_1$９０１および１６ビットのデータブロック$I_2$９０２と、を混ぜ合わせる。具体的には、図９に示すように、

「$M_1\rightarrow D_1$、$I_1\rightarrow D_2$、$M_2\rightarrow D_3$、$I_2\|I_2\|I_2\|I_2\rightarrow D_4$、$M_3\rightarrow D_5$、$M_4\rightarrow D_6$、$M_5\rightarrow D_7$、$M_6\rightarrow D_8$、$M_7\rightarrow D_9$、・・・・・」

といったデータの置き換えが行われる。

【０１８３】この置き換えた結果としてのＮ＋２個の６４ビットデータブロック$D_1$２１０、$D_2$３１１、$D_3$２１２、$D_4$２１３、$D_5$２１４、・・・がこの順番で連なったものが、中間拡大データ２１５として出力される。

【０１８４】この中間拡大データ２１５の長さは、Ｌ×６４ビットの整数倍となる。

【０１８５】単射拡大部８０３は、当該単射拡大部８０３に入力される拡大データ１０７に対して、換字・転置処理を行いながら単射拡大を行うという点で、図１に示す単射拡大部１０５と同様である。ただし、単射拡大部８０３からの出力（中間出力）を８０ビットのデータとするために、具体的な処理が異なってくる。

【０１８６】図１０は単射拡大部８０３での処理の一例を説明するためのフロー図である。

【０１８７】まず、ステップ１００２において、８０ビットの初期値８０２を入力する。そして、これをＨに設定する。

【０１８８】次に、ステップ１００３において、$q=1$に設定する。

【０１８９】次に、ステップ１００４において、図８に示す拡大データ１０７のうち、ｑ番目のフレームE_qを入力する。

【０１９０】次に、ステップ１００５において、フレームE_qに対し、Ｈをパラメータとして単射処理を施す。

【０１９１】次に、ステップ１００６において、ステップ１００５で得たデータに対し、Ｈをパラメータとして、その長さを６４ビットから８０ビットへ拡大するよ

うな単射拡大処理を施す。そして、得た８０ビットのデータをHに設定する。

【０１９２】次に、ステップ１００７では、q番目のフレームEqが、拡大データ１０７を構成するフレームE1、E2・・・のうちの最後のフレームに相当するか否かを判断する。最後のフレームである場合は、ステップ１００９へ移行し、そうでない場合は、ステップ１００８へ移行する。

【０１９３】ステップ１００８では、qの値を１つインクリメント（q＝q＋１）し、その後、ステップ１００４へ戻る。

【０１９４】ステップ１００９では、ステップ１００６で設定した８０ビットのデータHが、拡大データ１０７を構成するフレームのうちの最後のフレームに対するものであるので、このHをハッシュ値Hash８０４として出力する。

【０１９５】次に、図１０に示すステップ１００５での単射処理、およびステップ１００６での単射拡大処理について説明する。

【０１９６】まず、図１０に示すステップ１００５での単射処理（６４ビット→６４ビットの単射）について説明する。

【０１９７】図１１は、図１０に示すステップ１００５における６４ビットデータから６４ビットデータへの単射処理の一例を説明するためのフロー図である。

【０１９８】まず、ステップ１１０２において、図１０に示すステップ１００４で入力された６４ビットのフレームEqを、上位３２ビットのデータX1と下位３２ビットのデータY1とに分割する。

【０１９９】また、８０ビットのデータH（図１０において、ステップ１００４で入力されたフレームが１番目のフレームE1である場合は、ステップ１００２で設定した値、２番目以降のフレームE2・・・である場合は、直前に実施されたステップ１００６で設定した値）を、先頭から３２ビットデータH1、３２ビットデータH2、および１６ビットデータH3に分割する。

【０２００】次に、ステップ１１０３において、次式で示される処理を行うことで、X2、Y2を生成する。

【０２０１】$X_2=X_1+(Y_1+H_1)^2 \pmod{2^{32}}$

$Y_2=Y_1$

この結果、X2、Y2は、ともに３２ビットのデータとなる。なお、図１１のステップ１１０３において、細線の矢印は３２ビットデータの流れを示している。

【０２０２】次に、ステップ１１０４において、X2、Y2を出力し、その後、このフローを終了する。

【０２０３】上記のフローにより、３２ビットのデータX1および３２ビットのデータX1からなるフレームEqを、３２ビットのデータX2および３２ビットのデータY2の合計６４ビットのデータに変換することができる。また、X2、Y2が与えられたとき、$X_1=X_2-(Y_2+H_1)^2 \pmod{2^{48}}$、および、Y1＝Y2から、X1、Y1が一意的に定まる関係、すなわち単射とすることができる。

【０２０４】したがって、上記のフローにより、６４ビットデータから６４ビットデータへの単射を行うことができる。

【０２０５】ただし、図１０に示すステップ１００５での処理は、図１１に示すものに限定されるものではなく、６４ビットデータから６４ビットデータへの単射処理を行うものであればよい。

【０２０６】次に、図１０に示すステップ１００６での単射拡大処理（６４ビット→８０ビットの単射拡大）について説明する。

【０２０７】図１２は、図１０に示すステップ１００６における６４ビットデータから８０ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【０２０８】まず、ステップ１２０２において、図１１に示すフローで生成した３２ビットデータX2、Y2を入力する。また、図１１に示すステップ１１０２により８０ビットのデータHを分割することで生成した、３２ビットデータH1、H2と、１６ビットデータH3とを入力する。

【０２０９】次に、ステップ１２０３において、次式で示される処理を順番に行うことでX3、Y3を生成する。

【０２１０】$A=\mathrm{rot}_5(X_2+H_2+1)$

$B=A+X_2 \ \mathrm{eor}\ (H_3 \| H_3)+1$

$C=\mathrm{rot}_{13}(B+H_1)$

$D=C+(B \ \mathrm{or}\ H_2)+1$

$E=D^2+Y_2 \pmod{2^{48}}$

$F=X_2+H_3+E \pmod{2^{32}}$

$X_3=E+F+(H_1 \| H_2) \pmod{2^{48}}$

$Y_3=F$

ここで、 or は、ビット毎の論理和を示す。

【０２１１】上記の式で示される処理を順次行うことにより、X3は４８ビットのデータ、そして、Y3は３２ビットのデータとなる。なお、図１２のステップ１２０３において、太線の矢印は４８ビットデータの流れを、そして、細線の矢印は、３２ビットデータの流れを示している。

【０２１２】次に、ステップ１２０４において、X3、Y3を出力し、その後、このフローを終了する。

【０２１３】上記のフローにより、３２ビットデータX2および３２ビットデータY2の合計６４ビットのデータを、４８ビットのデータX3および３２ビットのデータY3の合計８０ビットのデータに拡大することができ、且つ、X3、Y3が与えられたとき、X2、Y2が一意的に定まる関係、すなわち単射とすることができる。

【０２１４】したがって、上記のフローにより、６４ビットデータから８０ビットデータへの単射拡大を行うことができる。

【０２１５】ただし、図１０に示すステップ１００６での処理は、図１２に示すものに限定されるものではなく、６４ビットデータから８０ビットデータへの単射拡大処理を行うものであればよい。

【０２１６】次に、本発明の第二実施形態について説明する。

【０２１７】図１３は本発明の第二実施形態であるデータ暗号化装置の機能構成を示す図である。このデータ暗号化装置は、第一実施形態のハッシュ値生成装置と同様に、パーソナルコンピュータやＩＣカードなどのマイクロプロセッサを備えた情報処理装置において、マイクロプロセッサに所定のプログラムを実行させることで実現可能である。また、１つのＬＳＩで実現させることも可能である。

【０２１８】図１３において、任意長のデータ鍵１３０２が、データ暗号化装置１３１１に入力されると、ハッシュ値生成装置１３０１において、２５６ビットのシステム鍵１３０３が初期値として与えられ、当該データ鍵１３０２に対する２５６ビットのハッシュ値が生成される。このハッシュ値がワーク鍵１３０４となる。

【０２１９】ワーク鍵１３０４は、８個の３２ビットデータW_1、W_2、・・・W_8に分割される。

【０２２０】ここで、ハッシュ値生成装置１３０２は、本発明の第一実施形態で説明したものでもよいし、従来の「ブロック暗号を利用したハッシュ関数」や「専用ブロック関数」であってもよい。

【０２２１】また、暗号化の対象となる１２８ビットの平文１３０５が、データ暗号化装置１３１１に入力されると、当該平文１３０５は、２個の６４ビットデータに分割されて、π関数処理部１３０６に入力される。

【０２２２】そして、８個の３２ビットデータW_1、W_2、・・・W_8を鍵として、２個の６４ビットのデータに変換（これについては後述する）された後、π関数処理部１３０７に入力され、π関数処理部１３０６での処理と同じ要領で、２個の６４ビットのデータに変換される。

【０２２３】その後、上記の処理がπ関数処理部１３０８～１３１３で順次行われ、これにより、π関数処理部１３１３から２個の６４ビットデータが出力される。この２個の６４ビットデータが結合され、１２８ビットの暗号文１３１０が生成される。

【０２２４】次に、図１３に示すπ関数処理部１３０６～１３１３での処理について説明する。

【０２２５】π関数処理１３０６～１３１３は、入力された２つの６４ビットデータに対し、ワーク鍵１３０４をパラメータとして、換字・転置処理を行う。ただし、本実施形態のπ関数処理１３０６～１３１３は、従来の換字・転置処理で用いられているＤＥＳなどの暗号化関数と異なり、３２ビット同士の乗算処理、および３２ビットの巡回シフト計算を含むようにしている。

【０２２６】図１４は、π関数処理１３０６～１３１３における処理の一例を説明するためのフロー図である。

【０２２７】まず、ステップ１４０２において、２個の６４ビットデータX_1、Y_1（π関数処理１３０６では、平文１２８ビットを２つに分割することで得られたデータ、π関数処理部１３０７～１３１３では、一つ前のπ関数処理部から出力されたデータ）を入力する。また、ワーク鍵１３０４を構成する８個の３２ビットデータW_1～W_8を入力する。

【０２２８】次に、ステップ１４０３において、ステップ１４０２で入力した６４ビットデータX_1を、上位３２ビットのデータX_Hと下位３２ビットのデータX_Lとに分割する。

【０２２９】次に、ステップ１４０４において、次式で示される処理を順番に行うことで、X_2、Y_2を生成する。

【０２３０】$A = X_L \text{ eor } W_1$

$B = X_H + W_2 + 1$

$C = A \cdot B$

$C' = (C \text{ eor } Y_2) + (W_3 \| W_4) + 1$

$C'_H \| C'_L = C'$

$D = \text{rot}_5(C'_L) \text{ eor } W_5$

$E = \text{rot}_{12}(C'_H) + W_6 + 1$

$F = D \| E$

$G = X_1 + F + 1$

$X_2 = C' + G + (W_7 \| W_8)$

$Y_2 = G$

上記の式で示される処理を順次行うことにより、X_2およびY_2は、ともに６４ビットのデータとなる。なお、図１４のステップ１４０４において、太線の矢印は６４ビットデータの流れを、そして、細線の矢印は、３２ビットデータの流れを示している。

【０２３１】次に、ステップ１４０５において、X_2、Y_2を出力し、その後、このフローを終了する。

【０２３２】上記のフローにより、６４ビットのデータX_1、Y_1の合計１２８ビットのデータを、換字・転置を行って、６４ビットのデータX_2、Y_2の合計１２８ビットのデータに変換することができる。

【０２３３】また、$X_1 \| Y_1 \rightarrow X_2 \| Y_2$への変換処理を全単射とすることができる。すなわち、出力$X_2 \| Y_2$から入力$X_1 \| Y_1$へ逆変換することが可能な関数であるπ^{-1}関数が存在する。

【０２３４】具体的には、次式で示される処理を順次行うことで、X_2、Y_2からX_1、Y_1を求めることができる。

【０２３５】$G = Y_2$

$C' = X_2 - G - (W_7 \| W_8)$

$C'_H \| C'_L = C'$

$D = \text{rot}_5(C'_L) \text{ eor } W_5$

$E = \text{rot}_{12}(C'_H) + W_6 + 1$

$F = D \| E$

$X_1 = G - F - 1$

$X_H \| X_L = X_2$

$A = X_L$ eor W_1

$B = X_H - W_2 - 1$

$C = A \cdot B$

$Y_1 = (C^{\prime}$　$(W_3 \cdot W_4)$　$1)$ eor C

上記の式で示される処理を順次行うことにより、X_1およびY_1を求めることができるので、このデータ暗号化装置１３１１が生成した暗号文１３１０は、π^{-1}関数を用いた逆変換により、元の平文１３０５に復号化することができる。

【０２３６】なお、図１３に示すπ関数処理部１３０６～１３１３での処理は、図１４に示すものに限定されるものではなく、全単射の換字・転置を行うに際して、３２ビット同士の乗算処理、および３２ビットの巡回シフト計算を含むようにしたものであればよい。

【０２３７】本実施形態の第二実施形態によれば、ワーク鍵１３０４を用いて平文１３０５を暗号化する過程において、換字・転置処理を行うπ関数処理部１３０６～１３１３において、３２ビットデータ同士の乗算処理を行っている（図１４に示すステップ１４０４での処理$C = A \cdot B$）。併せて、３２ビットデータに対する巡回シフト計算をも行っている（図１４に示すステップ１４０４での処理$D = rot_5 (C^{\prime}_L)$ eor W_5、$E = rot_{12} (C^{\prime}_H) + W_6 + 1$）。

【０２３８】このようにすることで、上述したように、換字・転置処理にＤＥＳなどの暗号化関数を用いた場合にくらべて、データ撹乱度の高い暗号文を、迅速に生成することができる。

【０２３９】なお、上記の第二実施形態では、図１３に示すように、π関数処理部での処理を８回行うもの（$n = 8$）について説明したが、本発明はこれに限定されるものではない。ｎの値を外部から与えるようにして、ｎを任意の正の整数に変更できるようにしてもよい。

【０２４０】次に、本発明の第三実施形態について説明する。

【０２４１】図１５は本発明の第三実施形態であるマスキング装置の機能構成を示した図である。ここで、マスキング装置とは、データをマスクする（覆い隠す）ためのデータを生成する装置のことである。マスキング装置で生成されたデータは、たとえばデータを暗号化するための鍵として用いることも可能である。

【０２４２】本実施形態のデータ暗号化装置は、第一実施形態のハッシュ値生成装置と同様に、パーソナルコンピュータやＩＣカードなどのマイクロプロセッサを備えた情報処理装置において、マイクロプロセッサに所定のプログラムを実行させることで実現可能である。また、１つのＬＳＩで実現させることも可能である。

【０２４３】図１５において、１２８ビットの共通鍵１５０２と、この共通鍵１５０２をＮ個繋げて構成された伸長データ１５０３とが入力されると、ハッシュ値生成装置１５０９において、乱数生成装置１５０８で生成された乱数１５２１と、入力された共通鍵１５０２とを繋ぎ合わせたデータに対する１２８ビットのハッシュ値が生成される。このハッシュ値が、伸長データ１５０３の最初の１２８ビットデータである第一区分１５０４を暗号化するためのワーク鍵１５２４となる。また、乱数生成装置１５０３で生成された乱数１５２１が、マスキングデータ１５２０の最初のデータとなる。

【０２４４】ここで、ハッシュ値生成装置１３０２は、本発明の第一実施形態で説明したものでもよいし、従来の「ブロック暗号を利用したハッシュ関数」や「専用ブロック関数」であってもよい。

【０２４５】一方、π関数処理部１５１３において、ワーク鍵１５２４の一部をパラメータとして、伸長データ１５０３の第一区分１５０４が換字・転置されて、１２８ビットのデータに変換される。その後、π関数処理部１５１４において、ワーク鍵１５２４の一部をパラメータとして、π関数処理部１５１３で生成された１２８ビットのデータが換字・転置されて、１２８ビットのデータに変換される。このデータが、マスキングデータ１５２０の２番目のデータ$g_1$１５２２となる。また、π関数処理部１５１３で生成された１２８ビットのデータが、伸長データ１５０３の２番目の１２８ビットデータである第二区分１５０５を暗号化するためのワーク鍵１５２５となる。

【０２４６】また、π関数処理部１５１８において、ワーク鍵１５２５の一部をパラメータとして、伸長データ１５０３の第二区分１５０５が換字・転置されて、１２８ビットのデータに変換される。その後、π関数処理部１５１９において、ワーク鍵１５２５の一部をパラメータとして、π関数処理部１５１８で生成された１２８ビットのデータが換字・転置されて、１２８ビットのデータに変換される。このデータが、マスキングデータ１５２０の３番目のデータ$g_2$１５２３となる。また、π関数処理部１５１８で生成された１２８ビットのデータが、伸長データの３番目の１２８ビットデータである第三区分（図示せず）を暗号化するためのワーク鍵１５２６となる。

【０２４７】上記の処理を伸長データを構成する全ての区分（１２８ビットデータ）に対して行うことで、マスキングデータ１５２０を生成する。

【０２４８】ここで、図１５に示すマスキング装置１５０１において、π関数処理部１５１３、１５１４、１５１８、１５１９、・・・は、図１３に示す第二実施形態で用いたπ関数処理部１３０６～１３１３と同じものである。

【０２４９】したがって、本発明の第三実施形態によれば、換字・転置処理にＤＥＳなどの暗号化関数を用いた

場合にくらべて、データ攪乱度の高いマスクデータを、迅速に生成することができる。

【０２５０】また、本発明の第三実施形態において、マスキングデータ１５２０は、共通鍵１５０２を伸長させたデータと見ることができる。また、マスキングデータ１５２０は、共通鍵１５０２によって伸長データ１５０３を暗号化したデータと見ることができる。すなわち、図１５に示す処理と逆の処理を行うことで、共通鍵１５０２およびマスキングデータ１５２０から、伸長データ１５０３を復号することができる。

【０２５１】なお、本発明の第三実施形態では、共通鍵１５０２の長さを１２８ビットとしたが、本発明はこれに限定されない。また、伸長データ１５０３の各区分１５０４、１５０５、・・・の各々に対して、π関数処理部での処理を２回行うようにしたが、本発明はこれに限定されるものではない。

【０２５２】次に、本発明の第四実施形態について説明する。

【０２５３】本実施形態は、電子メールなどのデータ暗号化システムに関するものであり、データ暗号化装置と、データ復号化装置とを含む。なお、以下に説明するデータ暗号化装置およびデータ復号化装置は、第一実施形態のハッシュ値生成装置と同様に、パーソナルコンピュータやＩＣカードなどのマイクロプロセッサを備えた情報処理装置において、マイクロプロセッサに所定のプログラムを実行させることで実現可能である。また、１つのＬＳＩで実現させることも可能である。

【０２５４】まず、データ暗号化装置について説明する。

【０２５５】図１６は、本発明の第四実施形態であるデータ暗号化システムを構成するデータ暗号化装置の機能構成を示す図である。

【０２５６】図１６に示すデータ暗号化装置１６０１には、楕円曲線暗号におけるパラメータであるベースポイントＰ１６０２、公開鍵Ｑ１６０３、および、平文１６０４が入力される。

【０２５７】ここで、楕円曲線暗号とは、次式、

$y^2 = x^3 + ax + b$

で表される楕円曲線上の２点（x_1，y_1）、（x_2，y_2）の加算（x_1，y_1）＋（x_2，y_2）や、整数倍演算ｋ（x_1，y_1）などを定義することにより生成される公開鍵暗号のことである。

【０２５８】ベースポイントＰ１６０２や公開鍵Ｑ１６０３も、上記の楕円曲線上の点であり、後で述べる秘密鍵ｄ１８０２との間で次の関係を満たす。

【０２５９】Ｑ＝ｄＰ

データ暗号化装置１６０１に入力されたベースポイントＰは１６０２は、乱数生成装置１６０７で生成された乱数ｋとともに、整数倍演算部１６０８へ入力される。これを受けて、整数倍演算部１６０８は、次式で示される処理を実行することで、データＲ１６１７を生成する。

【０２６０】Ｒ＝ｋＰ

このデータＲ１６１７は、暗号文１６１６の最初のデータとなる。

【０２６１】また、データ暗号化装置１６０１に入力された公開鍵Ｑ１６０３は、乱数生成装置１６０７で生成された乱数ｋとともに、整数倍演算部１６０９へ入力される。これを受けて、整数倍演算部１６０９は、次式で示される処理を実行することで、上記の楕円曲線上の点（ｘ，ｙ）を生成する。

【０２６２】（ｘ，ｙ）＝ｋＱ

また、データ暗号化装置１６０１に入力された平文１６０４のうち、最初のＮビットデータ１６０５は、圧縮・暗号化部１６１２に入力される。これを受けて、圧縮・暗号化部１６１２は、ハッシュ値生成装置１６１１で生成されたハッシュ値を鍵として、最初のＮビットデータ１６０５の圧縮・暗号化処理を行う。これによりデータ$C_1$１６１８を生成する。このデータ$C_1$１６１８は、暗号文１６１６の２番目のデータとなる。

【０２６３】なお、ハッシュ値生成装置１６１１は、番号生成部１６１０で生成されたシーケンス番号「１」と、整数倍演算部１６０９で生成した（ｘ，ｙ）のうちの数値ｘとを接続することで得られるデータに対するハッシュ値を生成する。

【０２６４】また、データ暗号化装置１６０１に入力された平文１６０４のうち、２番目のＮビットデータ１６０６は、圧縮・暗号化部１６１５に入力される。これを受けて、圧縮・暗号化部１６１５は、ハッシュ値生成装置１６１４で生成されたハッシュ値を鍵として、２番目のＮビットデータ１６０６の圧縮・暗号化処理を行う。これによりデータ$C_2$１６１９を生成する。このデータ$C_2$１６１９は、暗号文１６１６の３番目のデータとなる。

【０２６５】なお、ハッシュ値生成装置１６１４は、番号生成部１６１３で生成されたシーケンス番号「２」と、整数倍演算部１６０９で生成した（ｘ，ｙ）のうちの数値ｘとを接続することで得られるデータに対するハッシュ値を生成する。

【０２６６】上記の処理を平文１６０４を構成する全てのＮビットデータに対して行うことで、暗号文１６１６を生成する。

【０２６７】なお、図１６に示すデータ暗号化装置１６０１において、ハッシュ値生成装置１６１１、１６１４、・・・は、本発明の第一実施形態で説明したものでもよいし、従来の「ブロック暗号を利用したハッシュ関数」や「専用ブロック関数」であってもよい。

【０２６８】次に、図１６に示す圧縮・暗号化部１６１２、１６１５、・・・での処理について説明する。

【０２６９】図１７は、図１６に示す圧縮・暗号化部１６１２、１６１５、・・・の機能構成を示す図である。

【０２７０】ここで、Ｎビットデータとは、図１６において、平文１６０４を構成するＮビットデータ１６０５、１６０６、・・・に相当する。また、データＣは、図１６において、圧縮・暗号化部１６１２、１６１５、・・・が生成するデータ$C_1$１６１８、$C_2$１６１９、・・・を示す。さらに、鍵１７０５とは、図１６において、対応するハッシュ値生成装置１６１１、１６１４、・・・が生成したハッシュ値に相当する。

【０２７１】図１７において、鍵１７０５が入力されると、拡張部１７０６はこの鍵１７０５を受け取ってコピーを複数個生成し、これ等を繋ぎ合わせてワーク鍵１７２３を生成する。

【０２７２】また、Ｎビットデータの最初のデータ区分である第１区分１７０３は、圧縮処理部１７０７において、ワーク鍵１７２３の一部をパラメータとして、ハフマン圧縮などにより圧縮（換字）処理が施される。そして、その結果が、１２８ビットの圧縮データ１７０８および端数データ１７０６として出力される。

【０２７３】１２８ビットの圧縮データ１７０８は、π関数処理部１７１０において、ワーク鍵１７２３の一部をパラメータとして、換字・転置処理が施され、１２８ビットのデータに変換される。その後、π関数処理部１７１１において、ワーク鍵１７２３の一部をパラメータとして、さらに換字・転置処理が施されて、１２８ビットのデータに変換される。このデータが、生成すべきデータＣの最初のデータ$b_1$１７２０となる。また、π関数処理部１７１０で生成された１２８ビットのデータは、拡張部１７１２に入力され、複数コピーされる。そして、これ等が繋ぎ合わされて、Ｎビットデータの２番目のデータ区分である第２区分１７０４を暗号化するためのワーク鍵１７２２となる。

【０２７４】また、Ｎビットデータの２番目のデータ区分である第２区分１７０４は、圧縮処理部１７１３において、ワーク鍵１７２２の一部をパラメータとして、ハフマン圧縮などにより圧縮（換字）処理が施される。そして、その結果が、圧縮データ１７１４および端数データ１７１５として出力される。ここで、圧縮データ１７１４は、第１区分１７０３を圧縮（換字）処理した際に生成された端数データ１７０９との合計ビット長が、１２８ビットとなるように生成される。

【０２７５】圧縮データ１７１５は、第１区分１７０３を圧縮（換字）処理した際に生成された端数データ１７０９とを繋ぎ合わされて、１２８ビットのデータになる。その後、π関数処理部１７１６において、ワーク鍵１７２２の一部をパラメータとして、換字・転置処理が施され、１２８ビットのデータに変換される。さらにその後、π関数処理部１７１８において、ワーク鍵１７２２の一部をパラメータとして、さらに換字・転置処理が施されて、１２８ビットのデータに変換される。このデータが、生成すべきデータＣの２番目のデータ$b_2$１７２１となる。また、π関数処理部１７１６で生成された１２８ビットのデータは、拡張部１７１７に入力され、複数コピーされる。そして、これ等が繋ぎ合わされて、Ｎビットデータの３番目のデータ区分を暗号化するためのワーク鍵となる。

【０２７６】上記の処理をＮビットデータを構成する全ての区分に対して行うことで、対応するデータＣを生成する。

【０２７７】ここで、図１７に示すπ関数処理部１７１０、１７１１、１１７１６、１７１８、・・・は、図１３に示す第二実施形態で用いたπ関数処理部１３０６、１３０７、１３０８、・・・１３１３と同じものである。

【０２７８】本発明の第四実施形態を構成するデータ暗号化装置では、楕円暗号とハッシュ値生成装置とを組み合わせることで、データの暗号化を行っている。くわえて、圧縮・暗号化部として、第二実施形態で用いたπ関数処理部を利用している。

【０２７９】したがって、本発明の第四実施形態を構成するデータ暗号化装置によれば、従来のＲＳＡ（Rivest, Shamir, Adleman）のような公開鍵暗号方式のデータ暗号化装置に比べて、長いデータに対し、データ攪乱度の高い暗号文を、迅速に生成することができる。

【０２８０】なお、このデータ暗号化装置は、公開鍵Ｑ１６０３を用いて、平文１６０４を暗号文１６１６に変換するという点において、従来のＲＳＡの公開鍵暗号方式を用いたデータ暗号化装置と共通する。しかしながら、本発明の第四実施形態を構成するデータ暗号化装置では、上述したように、π関数処理部を用いて換字・転置処理を行っている点で従来のものと異なる。

【０２８１】また、圧縮処理部１７０７、１７１３、・・・において、ハフマン圧縮などにより圧縮（換字）処理を行っているので、このハフマン圧縮などにより圧縮することができる通常の平文に対しては、暗号文１６１６の長さが平文１６０４の長さよりも短くなる。この点でも、従来のものと異なる。

【０２８２】ところで、図１７に示す圧縮・暗号化部１６１２、１６１５、・・・では、Ｎビットデータを構成する各区分（第１区分１７０３、第２区分１７０４、・・・）に対して、π関数処理部での変換処理を２回行うようにしている。しかしながら、本発明はこれに限定されるものではない。

【０２８３】次に、データ復号化装置について説明する。

【０２８４】図１８は、本発明の第四実施形態であるデータ暗号化システムを構成するデータ復号化装置の機能構成を示す図である。

【０２８５】図１８に示すデータ復号化装置１８０１には、楕円曲線暗号におけるパラメータである秘密鍵ｄ１８０２、および、暗号文１８０３が入力される。

【０２８６】データ復号化装置１８０１に入力された秘密鍵ｄ１８０２は、同じくデータ復号化装置１８０１に入力された暗号文１８０３のうちの最初のデータＲ１８０４（これは、図１６におけるデータＲ１６１７に相当）とともに、整数倍演算部１８０７へ入力される。これを受けて、整数倍演算部１８０７は、次式で示される処理を実行することで、楕円曲線上の点（ｘ、ｙ）を生成する。

【０２８７】（ｘ，ｙ）＝ｋＲ

また、データ復号化装置１８０１に入力された暗号文１８０３のうち、２番目のデータ$C_1$１８０５は、復号・伸長化部１８１０に入力される。これを受けて、復号・伸長化部１８１０は、ハッシュ値生成装置１８０９で生成されたハッシュ値を鍵として、２番目のデータ$C_1$１８０５の復号・伸長化処理を行う。これにより、Ｎビットのデータを生成する。このデータは、平文１８１４の最初のＮビットデータ１８１５となる。

【０２８８】なお、ハッシュ値生成装置１８０９は、番号生成部１８０８で生成されたシーケンス番号「１」と、整数倍演算部１８０７で生成した（ｘ，ｙ）のうちの数値ｘとを接続することで得られるデータに対するハッシュ値を生成する。

【０２８９】また、データ復号化装置１８０１に入力された暗号文１８０３のうち、３番目のデータ$C_2$１８０６は、復号・伸長化部１８１３に入力される。これを受けて、復号・伸長化部１８１３は、ハッシュ値生成装置１８１２で生成されたハッシュ値を鍵として、３番目のデータ$C_2$１８０６の復号・伸長化処理を行う。これにより、Ｎビットのデータを生成する。このデータは、平文１８１４の２番目のＮビットデータ１８１６となる。

【０２９０】なお、ハッシュ値生成装置１８１２は、番号生成部１８１１で生成されたシーケンス番号「２」と、整数倍演算部１８０７で生成した（ｘ，ｙ）のうちの数値ｘとを接続することで得られるデータに対するハッシュ値を生成する。

【０２９１】上記の処理を暗号文１８０３を構成する２番目のデータ$C_1$１８０５から最後のデータまでに対して行うことで、平文１８１４を生成する。

【０２９２】なお、図１８に示すデータ復号化装置１８０１において、ハッシュ値生成装置１８０９、１８１２、・・・は、図１６に示すものと同じである。

【０２９３】次に、図１８に示す復号・伸長化部１８１０、１８１３、・・・での処理について説明する。

【０２９４】図１９は、図１８に示す復号・伸長化部１８１０、１８１３、・・・の機能構成を示す図である。

【０２９５】ここで、データＣは、図１８において、暗号文１８０３の２番目以降のデータ$C_1$１８０５、$C_2$１８０６、・・・を示す。また、Ｎビットデータとは、図１８において、復号・伸長化部１８１０、１８１３、・・・が生成したＮビットデータ１８１５、１８１６、・・・に相当する。さらに、鍵１９０５とは、図１８において、対応するハッシュ値生成装置１８０９、１８１２、・・・が生成したハッシュ値に相当する。

【０２９６】また、π^{-1}関数処理部１９０７、１９１０、１９１４、１９１６、・・・は、図１７に示すπ関数処理部１７１０、１７１１、１７１６、１７１８、・・・での処理と逆関数の関係にある処理を行う。すなわち、π^{-1}関数は、π関数の逆関数である。

【０２９７】同じパラメータをπ関数とπ^{-1}関数に設定した場合、データｍをπ関数によって変換して得られるデータπ（ｍ）を、さらに、π^{-1}関数によって変換すると、元のデータｍに戻る。すなわち、π^{-1}関数はπ関数と次式のような関係にある。

【０２９８】$m=\pi^{-1}(\pi(m))$

また、伸長処理部１９１１、１９１７、・・・は、図１７に示す圧縮処理部１７０７、１７１３、・・・での処理と逆変換の関係にある処理を行う。

【０２９９】同じパラメータを圧縮処理部と伸長処理部に設定した場合、データｍを圧縮（換字）処理によって変換して得られるデータを、さらに、伸長（換字）処理によって変換すると、元のデータｍに戻る。

【０３００】図１９において、鍵１９０５が入力されると、拡張部１９０６はこの鍵１９０５を受け取ってコピーを複数個生成し、これ等を繋ぎ合わせてワーク鍵１９２３を生成する。

【０３０１】また、データＣの最初の１２８ビットデータ$b_1$１９０３は、π^{-1}関数処理部１９０７において、ワーク鍵１９２３の一部をパラメータとして、換字・転置処理が施されて１２８ビットデータ変換された後、π^{-1}関数処理部１９１０において、ワーク鍵１９２３の一部をパラメータとして、さらに換字・転置処理が施されて、１２８ビットデータに変換される。

【０３０２】π^{-1}関数処理部１９１０の出力結果は、伸長処理部１９１１において、ワーク鍵１９２３の一部をパラメータとして、伸長（換字）処理が施される。そして、その結果が、１２８ビットの伸長データ１９１２およびその端数データ１９１３として出力される。この１２８ビットの伸長データが、生成すべきＮビットデータのうちの第１区分のデータ１９２１となる。また、π^{-1}関数処理部１９０７で生成された１２８ビットのデータは、拡張部１９０９に入力され、複数コピーされる。そして、これ等が繋ぎ合わされて、データＣの２番目の１２８ビットデータ$b_2$１９０４を復号化するためのワーク鍵１９２４となる。

【０３０３】また、データＣの２番目の１２８ビットデータ$b_2$１９０４は、π^{-1}関数処理部１９１４において、ワーク鍵１９２４の一部をパラメータとして、換字・転置処理が施されて１２８ビットデータに変換された後、π^{-1}関数処理部１９１６において、ワーク鍵１９２４の一部をパラメータとして、さらに換字・転置処理が

施されて、１２８ビットデータに変換される。

【０３０４】π^{-1}関数処理部１９１６の出力結果は、伸長処理部１９１７において、ワーク鍵１９２４の一部をパラメータとして、伸長（換字）処理が施される。そして、その結果が、伸長データ１９１８およびその端数データ１９１９として出力される。

【０３０５】ここで、伸長データ１９１８は、データ$b_1$１９０３を伸長（換字）処理した際に生成された端数データ１９１３との合計ビット長が、１２８ビットとなるように生成される。この伸長データ１９１８と端数データ１９１３とを繋いだ合計１２８ビットのデータが、生成すべきＮビットデータのうちの第２区分のデータ１９２２となる。また、π^{-1}関数処理部１９１４で生成された１２８ビットのデータは、拡張部１９１５に入力され、複数コピーされる。そして、これ等が繋ぎ合わされて、データＣの３番目の１２８ビットデータを復号化するためのワーク鍵となる。

【０３０６】上記の処理をデータＣを構成する全ての１２８ビットデータb_1、b_2、・・・に対して行うことでＮビットデータを生成する。

【０３０７】本発明の第四実施形態を構成するデータ復号化装置は、秘密鍵ｄ１８０２を用いて、暗号文１８０３を平文１８１４に復号変換するという点において、従来のＲＳＡの公開鍵暗号方式を用いたデータ復号化装置と共通する。しかしながら、本発明の第四実施形態を構成するデータ暗号化装置では、上述したように、π^{-1}関数処理部を用いて換字・転置処理を行っている点で従来のものと異なる。

【０３０８】また、伸長処理部１９１１、１９１７、・・・において、伸長（換字）処理を行っているので、このハフマン圧縮などで圧縮された暗号文に対して、復号化された平文１８１４は、そのデータ長が長くなる。この点でも、従来のものと異なる。

【０３０９】次に、上記の第四実施形態におけるデータ暗号化装置の変形例について説明する。

【０３１０】図２０は、図１６に示す本発明の第四実施形態におけるデータ暗号化装置の変形例の機能構成を示す図である。

【０３１１】図２０に示すデータ暗号化装置２００１には、楕円曲線暗号におけるパラメータであるベースポイントＰ２００２、公開鍵Ｑ２００３、および、平文２００４が入力される。

【０３１２】データ暗号化装置２００１に入力されたベースポイントＰ２００２は、乱数生成装置２００７で生成された乱数ｋとともに、整数倍演算部２００８へ入力される。これを受けて、整数倍演算部２００８は、次式で示される処理を実行することで、データＲ２０１７を生成する。

【０３１３】Ｒ＝ｋＰ

このデータＲ２０１７は、暗号文２０１６の最初のデータとなる。

【０３１４】また、データ暗号化装置２００１に入力された公開鍵Ｑ２００３は、乱数生成装置２００７で生成された乱数ｋとともに、整数倍演算部２００９へ入力される。これを受けて、整数倍演算部２００９は、次式で示される処理を実行することで、楕円曲線上の点（ｘ，ｙ）を生成する。

【０３１５】（ｘ，ｙ）＝ｋＱ

また、データ暗号化装置２００１に入力された平文２００４のうち、最初のＮビットデータ２００５は、圧縮・暗号化部２０１２に入力される。これを受けて、圧縮・暗号化部２０１２は、ハッシュ値生成装置２０１１で生成されたハッシュ値を鍵２０２０として、最初のＮビットデータ２００５の圧縮・暗号化処理を行う。これによりデータ$C_1$２０１８を生成する。このデータ$C_1$２０１８は、暗号文２０１６の２番目のデータとなる。

【０３１６】なお、ハッシュ値生成装置２０１１は、整数倍演算部２００９で生成した（ｘ，ｙ）のうちの数値ｘに対するハッシュ値を生成する。このハッシュ値が上記の鍵２０２０となる。

【０３１７】また、データ暗号化装置２００１に入力された平文２００４のうち、２番目のＮビットデータ２００６は、圧縮・暗号化部２０１５に入力される。これを受けて、圧縮・暗号化部２０１５は、ハッシュ値生成装置２０１４で生成されたハッシュ値を鍵２０２１として、２番目のＮビットデータ２００６の圧縮・暗号化処理を行う。これによりデータ$C_2$２０１９を生成する。このデータ$C_2$２０１９は、暗号文２０１６の３番目のデータとなる。

【０３１８】なお、ハッシュ値生成装置２０１４は、鍵２０２０に対するハッシュ値を生成する。このハッシュ値が上記の鍵２０２１となる。

【０３１９】上記の処理を平文２００４を構成する全てのＮビットデータに対して行うことで、暗号文２０１６を生成する。

【０３２０】なお、図１６に示すデータ暗号化装置１６０１において、圧縮・暗号化部２０１２、２０１５、・・・には、図１６に示す圧縮・暗号化部１６１２、１６１５、．．．と同じものが用いられる。

【０３２１】次に、本発明の第五実施形態について説明する。

【０３２２】本実施形態は、本発明の第四実施形態と同様、電子メールなどのデータ暗号化システムに関するものであり、データ暗号化装置と、データ復号化装置とを含む。なお、以下に説明するデータ暗号化装置およびデータ復号化装置は、第一実施形態のハッシュ値生成装置と同様に、パーソナルコンピュータやＩＣカードなどのマイクロプロセッサを備えた情報処理装置において、マイクロプロセッサに所定のプログラムを実行させることで実現可能である。また、１つのＬＳＩで実現させるこ

とも可能である。

【０３２３】まず、データ暗号化装置について説明する。

【０３２４】図２１は、本発明の第五実施形態であるデータ暗号化システムを構成するデータ暗号化装置の機能構成を示す図である

【０３２５】図２１に示すデータ暗号化装置２１０１には、楕円曲線暗号におけるパラメータであるベースポイントＰ２１０２、公開鍵$Q_1$２１０３、公開鍵$Q_2$２１０４、公開鍵$Q_3$２１０５、公開鍵$Q_4$２１２３、および、平文２１０６が入力される。

【０３２６】データ暗号化装置２１０１に入力されたベースポイントＰ２１０２は、乱数生成装置２１１３で生成された乱数ｋとともに、整数倍演算部２１２３へ入力される。これを受けて、整数倍演算部２１２３は、次式で示される処理を実行することで、データＲ２１０９を生成する。

【０３２７】Ｒ＝ｋＰこのデータＲ２１０９は、暗号文２１０８の最初のデータとなる。

【０３２８】また、データ暗号化装置２１０１に入力された公開鍵$Q_1$２１０３は、乱数生成装置２１１３で生成された乱数ｋとともに、整数倍演算部２１１４へ入力される。これを受けて、整数倍演算部２１１４は、次式で示される処理を実行することで、上記の楕円曲線上の点（x_1，y_1）を生成する。

【０３２９】（x_1，y_1）＝ｋQ_1

この（x_1，y_1）のうち、数値x_1は、その後、ハッシュ値生成装置２１１９に入力されて、ハッシュ値ｈ（x_1）に変換される。

【０３３０】同様に、データ暗号化装置２１０１に入力された公開鍵$Q_2$２１０４は、乱数生成装置２１１３で生成された乱数ｋとともに、整数倍演算部２１１５へ入力される。これを受けて、整数倍演算部２１１５は、次式で示される処理を実行することで、上記の楕円曲線上の点（x_2，y_2）を生成する。

【０３３１】（x_2，y_2）＝ｋQ_2

この（x_2，y_2）のうち、数値x_2は、その後、ハッシュ値生成装置２１２６に入力されて、ハッシュ値ｈ（x_2）に変換される。

【０３３２】また、同様に、データ暗号化装置２１０１に入力された公開鍵$Q_3$２１０５は、乱数生成装置２１１３で生成された乱数ｋとともに、整数倍演算部２１１６へ入力される。これを受けて、整数倍演算部２１１６は、次式で示される処理を実行することで、上記の楕円曲線上の点（x_3，y_3）を生成する。

【０３３３】（x_3，y_3）＝ｋQ_3

この（x_3，y_3）のうち、数値x_3は、その後、ハッシュ値生成装置２１２７に入力されて、ハッシュ値ｈ（x_3）に変換される。

【０３３４】さらに、同様に、データ暗号化装置２１０１に入力された公開鍵$Q_4$２１２３は、乱数生成装置２１１３で生成された乱数ｋとともに、整数倍演算部２１２４へ入力される。これを受けて、整数倍演算部２１２４は、次式で示される処理を実行することで、上記の楕円曲線上の点（x_4，y_4）を生成する。

【０３３５】（x_4，y_4）＝ｋQ_1

この（x_4，y_4）のうち、数値x_4は、その後、ハッシュ値生成装置２１２８に入力されて、ハッシュ値ｈ（x_4）に変換される。

【０３３６】しきい値ロジック部２１２５は、ハッシュ値生成装置２１２６、２１２７、２１２８で生成したハッシュ値ｈ（x_2）、ｈ（x_3）、ｈ（x_4）のうち、いずれか２つのハッシュ値が分かれば、ハッシュ値生成装置２１１９で生成したハッシュ値ｈ（x_1）を求めることができる条件式を満たす値$f_1$２１１０、$f_2$２１１１を生成する。この値$f_1$２１１０、$f_2$２１１１は、それぞれ、暗号文２１０８の２番目、３番目のデータとなる。

【０３３７】なお、このしきい値ロジック部２１２５の詳細については後述する。

【０３３８】データ暗号化装置２１０１に入力された平文２１０６のうち、最初のＮビットデータ２１０７は、圧縮・暗号化部２１２０に入力される。これを受けて、圧縮・暗号化部２１２０は、ハッシュ値生成装置２１１９で生成されたハッシュ値ｈ（x_1）を鍵２１２０として、最初のＮビットデータ２１０７の圧縮・暗号化処理（これについては後述する）を行う。これによりデータ$C_1$２１１２を生成する。このデータ$C_1$２１１２は、暗号文２１０８の４番目のデータとなる。また、鍵２１２０は、ハッシュ値生成装置２１２１に入力されて、平文２１０６の２番目のＮビットデータを暗号化するための鍵２１２２に変換される。

【０３３９】上記の処理を平文２１０６を構成する全てのＮビットデータに対して行うことで、暗号文２１０８１２を生成する。

【０３４０】なお、図２１に示すデータ暗号化装置２１０１において、ハッシュ値生成装置は、本発明の第一実施形態で説明したものでもよいし、従来の「ブロック暗号を利用したハッシュ関数」や「専用ブロック関数」であってもよい。

【０３４１】次に、図２１に示すしきい値ロジック部２１２５での処理について説明する。

【０３４２】上述したように、しきい値ロジック部２１２５は、ハッシュ値ｈ（x_2）、ｈ（x_3）、ｈ（x_4）のうちのいずれか２つのハッシュ値が分かれば、ハッシュ値ｈ（x_1）を求めることができる条件式を満たす値f_1、f_2を生成する。

【０３４３】図２２は、図２１に示すしきい値ロジック部２１２５の機能構成を示す図である。

【０３４４】図２２に示すように、しきい値ロジック部

２１２５には、５つのデータq_{1X}２２０２、ｈ（x_1）２２０３、ｈ（x_2）２２０４、ｈ（x_3）２２０５、ｈ（x_4）２２０６が入力される。

【０３４５】ここで、q_{1X}２２０２は、図２１において、公開鍵$Q_1$２１０３のｘ座標値である。また、ｈ（x_1）２２０３、ｈ（x_2）２２０４、ｈ（x_3）２２０５、ｈ（x_4）２２０６は、それぞれ、ハッシュ値生成装置２１１９、２１２６、２１２７、２１２８で生成されたハッシュ値である。

【０３４６】演算部２２０８は、次式で示される処理を実行することで、データ$f_1$２１１０を生成する。

【０３４７】$f_1 = g(q_{1X}, h(x_1), h(x_2), h(x_3), h(x_4))$

また、演算部２２１０は、次式で示される処理を実行することで、データ$f_2$２１１１を生成する。

【０３４８】$f_2 = g(h(q_{1X}), h(x_1), h(x_2), h(x_3), h(x_4))$

なお、上記の式f_1、f_2において、関数ｇは、$g(x, a_1, a_2, a_3, a_4) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \pmod n$で定義されものである。

【０３４９】したがって、演算部２２０８、２２１０で生成されたデータf_1、f_2は、q_{1X}、および、ｈ（x_1），ｈ（x_2），ｈ（x_3），ｈ（x_4）との間において、以下に示す四元連立方程式が成立する。

【０３５０】$f_1 = h(x_1) + h(x_2) \cdot q_{1X} + h(x_3) \cdot q_{1X}^2 + h(x_4) \cdot q_{1X}^3 \pmod n$

$f_2 = h(x_1) + h(x_2) \cdot h(q_{1X}) + h(x_3) \cdot h(q_{1X})^2 + h(x_4) \cdot h(q_{1X})^3 \pmod n$

よって、f_1、f_2、q_{1X}の値が既知である場合、ｈ（x_1），ｈ（x_2），ｈ（x_3），ｈ（x_4）のうちのいずれか二つの値が分かれば、他の二つの値も分かることになる（未知数２、方程式数２の連立方程式となるため）。

【０３５１】次に、図２１に示す圧縮・暗号化部２１２０、・・・での処理について説明する。

【０３５２】図２３は、図２１に示す圧縮・暗号化部２１２０、・・・の機能構成を示す図である。

【０３５３】ここで、Ｎビットデータとは、図２１において、平文２１０６を構成するＮビットデータ２１０７、・・・に相当する。また、データＣは、図２１において、圧縮・暗号化部２１２０、・・・が生成するデータ$C_1$２１１２、・・・を示す。さらに、鍵２３０９とは、図２１において、鍵２１２０、２１２２、・・・に相当する。

【０３５４】図２３において、鍵２３０９が入力されると、拡張部２３１０はこの鍵２３０９を受け取ってコピーを複数個生成し、これ等を繋ぎ合わせてワーク鍵２３１１を生成する。

【０３５５】また、Ｎビットデータの最初のデータ区分である第１区分２３０３は、圧縮処理部２３１２において、ワーク鍵２３１１の一部をパラメータとして、ハフマン圧縮などにより圧縮（換字）処理が施される。そして、その結果が、１２８ビットの圧縮データ２３１３およびその端数データ２３１４として出力される。

【０３５６】１２８ビットの圧縮データ２３１３は、π関数処理部２３１５において、ワーク鍵２３１１の一部をパラメータとして、ブロック暗号処理が施され、１２８ビットのデータに変換される。その後、π関数処理部２３１６において、ワーク鍵２３１１の一部をパラメータとして、さらにブロック暗号処理が施されて、１２８ビットのデータに変換される。このデータが、生成すべきデータＣの最初のデータ$b_1$２３０６となる。また、π関数処理部２３１５で生成された１２８ビットのデータは、拡張部２３１７に入力され、複数コピーされる。そして、これ等が繋ぎ合わされて、Ｎビットデータの２番目のデータ区分である第２区分２３０４を暗号化するためのワーク鍵２３１８となる。

【０３５７】また、Ｎビットデータの２番目のデータ区分である第２区分２３０４は、圧縮処理部２３１９において、ワーク鍵２３１８の一部をパラメータとして、ハフマン圧縮などにより圧縮（換字）処理が施される。そして、その結果が、圧縮データ２３２０および端数データ２３２１として出力される。ここで、圧縮データ２３２０は、第１区分２３０３を圧縮（換字）処理した際に生成された端数データ２３１４との合計ビット長が、１２８ビットとなるように生成される。

【０３５８】圧縮データ２３２０は、第１区分２３０３を圧縮（換字）処理した際に生成された端数データ２３１４とを繋ぎ合わされて、１２８ビットのデータになる。その後、π関数処理部２３２２において、ワーク鍵２３１８の一部をパラメータとして、ブロック暗号処理が施され、１２８ビットのデータに変換される。その後、π関数処理部２３２４において、ワーク鍵２３１８の一部をパラメータとして、さらにブロック暗号処理が施されて、１２８ビットのデータに変換される。このデータが、生成すべきデータＣの２番目のデータ$b_2$２３０７となる。

【０３５９】なお、第２区分２３０４を圧縮（換字）処理した際に生成された端数データ２３２１がｒビット（ｒ≧１）である場合、ハッシュ値生成装置２３２３は、鍵２３０９に対するハッシュ値を生成する。これを受けて、演算部２３２５は、ハッシュ値生成装置２３２３で生成されたハッシュ値の上位ｒビットと、端数データ２３２１との排他的論理的をとり、ｒビットのデータを生成する。このデータが、生成すべきデータＣの３番目のデータ$b_3$２３０８となる。

【０３６０】上記の処理により、Ｎビットデータに対する暗号データＣが生成される。

【０３６１】本発明の第五実施形態を構成するデータ暗号化装置では、平文２１０６の暗号化には、初期値とし

て、ハッシュ値ｈ（ｘ$_1$）を用いている。したがって、このデータ暗号化装置で暗号化された暗号文２１０８を復号するためには、ハッシュ値ｈ（ｘ$_1$）求めればよい。

【０３６２】ところで、このデータ暗号化装置では、しきい値ロジック部２１２５において、ハッシュ値ｈ（ｘ$_1$）、ｈ（ｘ$_2$）、ｈ（ｘ$_3$）、ｈ（ｘ$_4$）のうちのいずれか２つのハッシュ値が分かれば、ハッシュ値ｈ（ｘ$_1$）を求めることができる条件式を満たす値ｆ$_1$、ｆ$_2$を生成し、このｆ$_1$、ｆ$_2$を暗号文２１０８に付加している。

【０３６３】このため、暗号文２１０８を受信した者（すなわち、ｆ$_1$、ｆ$_2$を取得した者）は、ｈ（ｘ$_2$）、ｈ（ｘ$_3$）およびｈ（ｘ$_4$）のうちのいずれか２つ（ただし、図２２に示す例でいえば、さらにｑ$_{1x}$が必要）が分かれば、暗号化に用いたｈ（ｘ$_1$）を求めることができる。

【０３６４】よって、暗号文２１０８を復号は、公開鍵Ｑ$_1$と対になる秘密鍵ｄ$_1$を所有する（単独でハッシュ値ｈ（ｘ$_1$）を求めることができる）者のみならず、公開鍵Ｑ$_2$と対になる秘密鍵ｄ$_2$を所有する（単独でハッシュ値ｈ（ｘ$_2$）を求めることができる）者、公開鍵Ｑ$_3$と対になる秘密鍵ｄ$_3$を所有する（単独でハッシュ値ｈ（ｘ$_3$）を求めることができる）者、および、公開鍵Ｑ$_4$と対になる秘密鍵ｄ$_4$を所有する（単独でハッシュ値ｈ（ｘ$_4$）を求めることができる）者のうちの、いずれか二人が協力することでも行うことができる。

【０３６５】前者は、１ out of １の復号ロジックであり、後者は２ out of ３の復号ロジックである。本実施形態によれば、このように、復号可能となる受信者数のしきい値制御が可能となる。

【０３６６】すなわち、暗号文２１０８を秘密鍵ｄ$_1$、ｄ$_2$、ｄ$_3$、ｄ$_4$の所有者各々に対して同報通信（マルチキャスト）することで、秘密鍵ｄ$_1$の持ち主が単独で復号できるとともに、秘密鍵ｄ$_2$、ｄ$_3$、ｄ$_4$の持ち主３人のうちの２人が協力すれば復号可能なようにすることができる。

【０３６７】次に、図２１に示すデータ暗号化装置の変形例について説明する。

【０３６８】図２５は図２１に示すデータ暗号化装置の変形例の機能構成を示す図である。

【０３６９】ここで、図２１に示すデータ暗号化装置２１０１と同じ機能を有するものには、同じ符号を付している。

【０３７０】図２５に示すデータ暗号化装置２１０１ａは、整数倍演算部２１２３に代えて整数倍演算部２５０１を用いた点で、図２１に示すデータ暗号化装置２１０１と異なる。その他の構成は、図２１に示すものと同じである。

【０３７１】整数倍演算部２５０１は、ベースポイントＰ２１０２と、乱数生成装置２１１３で生成された乱数ｋとを受け取って、Ｒ＝ｋＰで示される処理を実行する点で、図２１に示す整数倍演算部２１２３と同じである。しかしながら、上記の式で求められるＲのうち、ｘ座標の値Ｒ$_x$２５０２のみを出力する点で異なる。

【０３７２】したがって、図２５に示すデータ暗号化装置２１０１ａでは、このデータＲ$_x$２５０２が、暗号文２１０８ａの最初のデータとなる。

【０３７３】図２５に示すデータ暗号化装置２１０１ａでは、図２１に示すデータ暗号化装置２１０１に比べて、生成される暗号文の長さが幾分短くなる。

【０３７４】すなわち、図２１に示すデータ暗号化装置２１０１では、整数倍演算部２１２３において、楕円曲線上の点Ｒ＝ｋＰを求め、求めたＲのｘ座標値Ｒ$_x$およびｙ座標値Ｒ$_y$を出力して、暗号文２１０８の最初のデータＲ２１０９としていた。

【０３７５】これに対し、図２５に示すデータ暗号化装置２１０１ａでは、整数倍演算部２５０１において、楕円曲線上の点Ｒ＝ｋＰを求め、求めたＲのうちのｘ座標値Ｒ$_x$のみを出力して、暗号文２１０８ａの最初のデータＲ$_x$２５０２としている。

【０３７６】したがって、図２５に示すデータ暗号化装置２１０１ａで生成された暗号文２１０８ａは、図２１に示すデータ暗号化装置２１０１で生成された暗号文２１０８よりも、ｙ座標値Ｒ$_y$のデータ分（たとえば、１６０ビット）だけ短くなる。

【０３７７】次に、データ復号化装置について説明する。

【０３７８】図２６は、本発明の第五実施形態であるデータ暗号化システムを構成するデータ復号化装置の機能構成を示す図である。このデータ復号化装置は、図２５に示すデータ暗号化装置２１０１ａで生成された暗号文２１０８ａを復号するためのものである。

【０３７９】データ復号化装置は、図２６に示すように、データ回復機関２６０７と、鍵保管機関Ａ２６０８と、鍵保管機関Ｂ２６０９とで構成される。

【０３８０】図２６に示すデータ回復機関２６０７に、公開鍵Ｑ$_1$２１０３と、暗号文２１０８ａとが入力されると、演算部２６１１は、暗号文２１０８ａの最初のデータＲ$_x$２５０２との間で、次の楕円曲線方程式を満足するｙ座標値Ｒ$_y$を求める。

【０３８１】$R_y^2 = R_x^3 + a \cdot R_x + b$ 通常、この方程式の解Ｒ$_y$は２つ存在する。一方の解をｒとすると、他方は－ｒとなる（ただし、楕円曲線として$y^2 + xy = x^3 + ax + b$を用いる場合には、一方の解をｒとすると、他方はＲ$_x$＋ｒとなる）。

【０３８２】どちらか任意の一方をとり、他方を無視する。ここでは、ｒの方をとるものとする。演算部２６１１は、Ｒ＝（Ｒ$_x$，ｒ）を出力する。

【０３８３】次に、データ回復機関２６０７は、演算部２６１１から出力された値Ｒを鍵保管機関Ａ２６０８、

Ｂ２６０９に各々送信する。

【０３８４】これを受けて、鍵保管機関Ａ２６０８、以下に示す処理を順次行う。

【０３８５】　自己が格納している秘密鍵$d_2$２６１８（公開鍵$Q_2$２１０４と対）を読み出す。

【０３８６】　演算部２６１９において、データ回復機関２６０７から送られてきた値Ｒとの間で、次式を満たす楕円曲線上の点（x_2，y_2）を求める。

【０３８７】（x_2，y_2）＝d_2Ｒ

ハッシュ値生成装置２６２０において、演算部２６１９で求めた（x_2，y_2）のうちのｘ座標値x_2のハッシュ値ｈ（x_2）を生成する。

【０３８８】　ハッシュ値生成装置２６２０で生成したハッシュ値ｈ（x_2）を、データ回復機関２６０７へ送信する。

【０３８９】また、鍵保管機関Ｂ２６０９は、以下に示す処理を順次行う。

【０３９０】　自己が格納している秘密鍵$d_3$２６２１（公開鍵$Q_3$２１０５と対）を読み出す。

【０３９１】　演算部２６２２において、データ回復機関２６０７から送られてきた値Ｒとの間で、次式を満たす楕円曲線上の点（x_3，y_3）を求める。

【０３９２】（x_3，y_3）＝d_3Ｒ

ハッシュ値生成装置２６２３において、演算部２６２２で求めた（x_3，y_3）のうちのｘ座標値x_3のハッシュ値ｈ（x_3）を生成する。

【０３９３】　ハッシュ値生成装置２６２３で生成したハッシュ値ｈ（x_3）を、データ回復機関２６０７へ送信する。

【０３９４】上記のような処理の結果、得られたハッシュ値ｈ（x_2）、ｈ（x_3）は以下に示すような特徴を有する。

【０３９５】かりに、鍵保管機関Ａ２６０８、Ｂ２６０９に各々送信されるデータが、Ｒ＝（R_x，ｒ）ではなく、Ｒ´＝（R_x，－ｒ）であったとする。

【０３９６】この場合、鍵保管機関Ａ２６０８の演算部２９１９では、（x_2´，y_2´）＝d_2Ｒ´の計算が行われことになる。

【０３９７】しかし、楕円曲線上の演算の性質より、（x_2´，y_2´）＝（x_2，－y_2）となる。

【０３９８】すなわち、楕円曲線$y^2=x^3+ax+b$を用いた演算では、－（ｘ，ｙ）＝（ｘ，－ｙ）が成立する。このとき、（x_2，y_2）＝d_2（R_x，ｒ）であったとすると、d_2（R_x，－ｒ）＝d_2（－（R_x，ｒ））＝－d_2（R_x，ｒ）＝（x_2，－y_2）となる。

【０３９９】また、楕円曲線$y^2+xy=x^3+ax+b$を用いた演算では、－（ｘ，ｙ）＝（ｘ，ｘ＋ｙ）が成立する。このとき、（x_2，y_2）＝d_2（Ｒｘ，ｒ）であったとすると、d_2（R_x，R_x＋ｒ）＝d_2（－（R_x，ｒ））＝－d_2（R_x，ｒ）＝（x_2，x_2＋y_2）となる。

【０４００】したがって、ｈ（x_2´）＝ｈ（x_2）となる。

【０４０１】つまり、Ｒ＝（Ｒｘ，ｒ）を入力しても、あるいはＲ´＝（Ｒｘ，－ｒ）を入力しても、鍵保管機関Ａ２６０８が出力するハッシュ値ｈ（x_2）は変化しない。

【０４０２】同様に、Ｒ＝（Ｒｘ，ｒ）を入力しても、あるいはＲ´＝（Ｒｘ，－ｒ）を入力しても、鍵保管機関Ｂ２６０９が出力するハッシュ値ｈ（x_3）も変化しない。

【０４０３】データ回復機関２６０７の演算部２６１１において、方程式の解Ｒｙが２つ存在するにもかかわらず、どちらか任意の一方のみを、鍵保管機関Ａ２６０８および鍵保管機関Ｂ２６０９に送信したのは、このような理由による。

【０４０４】次に、データ回復機関２６０７は、鍵保管機関Ａ２６０８および鍵保管機関Ｂ２６０９から各々ハッシュ値ｈ（x_2）、ｈ（x_3）を受け取ると、しきい値逆算ロジック部２６１２による処理を開始する。

【０４０５】しきい値逆算ロジック部２６１２は、まず、ハッシュ値ｈ（x_2）、ｈ（x_3）と、公開鍵$Q_1$２１０３のｘ座標値q_{1x}と、暗号文２１０８ａの２番目および３番目のデータ$f_1$２１１０、$f_2$２１１１と、を受け取る。

【０４０６】そして、以下に示す２元連立方程式を満たすｈ（x_1）、ｈ（x_4）を生成する。

【０４０７】$f_1=h(x_1)+h(x_2)\cdot q_{1x}+h(x_3)\cdot q_{1x}^2+h(x_4)\cdot q_{1x}^3 \pmod n$

$f_2=h(x_1)+h(x_2)\cdot h(q_{1x})+h(x_3)\cdot h(q_{1x})^2+h(x_4)\cdot h(q_{1x})^3 \pmod n$

ここで、上記の２元連立方程式は、図２１および図２５に示すハッシュ値生成装置で生成されたハッシュ値ｈ（x_1）、ｈ（x_2）、ｈ（x_3）、ｈ（x_4）の値を未知とする４元連立方程式のうち、ｈ（x_2）とｈ（x_3）とを既知とした場合に相当する。

【０４０８】図２１および図２５に示すハッシュ値生成装置では、乱数生成装置で生成した乱数ｋを用いて（x_2，y_2）＝ｋQ_2を求め、このうちのｘ座標の数値x_2に対するハッシュ値をｈ（x_2）としている。同様に、乱数ｋを用いて（x_3，y_3）＝ｋQ_3を求め、このうちのｘ座標の数値x_3に対するハッシュ値をｈ（x_3）としている。

【０４０９】これに対し、図２６に示すハッシュ値生成装置では、上記と同じ乱数ｋを用いてＲ＝ｋＰにより求められたＲ（あるいは、このＲとｙ座標の符号だけ異なるＲ´）を用いて、d_2Ｒ（あるいは、d_2Ｒ´）を求め、このうちのｘ座標の数値x_2（あるいは、x_2´）対するハッシュ値をｈ（x_2）としている。同様に、d_3Ｒ（あるいは、d_3Ｒ´）を求め、このうちのｘ座標の数値x_3（あるいは、x_3´）対するハッシュ値をｈ

（ｘ3）としている。

【０４１０】ここで、楕円曲線暗号における秘密鍵と公開鍵との関係から、$Q_2 = d_2P$、$Q_3 = d_3P$が成立する。したがって、

$d_2R = d_2kP = kd_2P = kQ_2 = (x_2, y_2)$

$d_3R = d_3kP = kd_3P = kQ_3 = (x_3, y_3)$

となる。また、上述した理由により、d_2R、d_3Rのｘ座標は、それぞれｘ2、ｘ3となる。

【０４１１】したがって、図２６に示すハッシュ値生成装置で生成されたハッシュ値ｈ（ｘ2）、ｈ（ｘ3）と、図２１および図２５に示すハッシュ値生成装置で生成されたハッシュ値ｈ（ｘ2）、ｈ（ｘ3）は各々一致するので、図２６に示すしきい値逆ブロック部２６１２で求めたハッシュ値ｈ（ｘ1）、ｈ（ｘ4）は、図２１および図２５に示すハッシュ値生成装置で生成されたハッシュ値ｈ（ｘ1）、ｈ（ｘ4）と各々一致する。

【０４１２】復号処理部２６１３は、しきい値逆ブロック部２６１２で求めたハッシュ値ｈ（ｘ1）を鍵２６１３として、暗号文２１０８ａの４番目のでデータC_{12}１１２に対して復号・伸長処理を行う。これにより、平文２１０６の最初のＮビットデータ２１０７を生成する。

【０４１３】さらに、復号処理部２６１７は、ハッシュ値生成装置２６１５において、鍵２６１３を基に生成されたハッシュ値を新しい鍵２６１６として、暗号文２１０８ａの５番目のデータに対して復号・伸長処理を行う。これにより、平文２１０６の２番目のＮビットデータを生成する。

【０４１４】上記の処理を、暗号文２１０８ａを構成する最後のデータまで、順次繰り返すことで、平文２１０６を復元することができる。

【０４１５】なお、本発明の第五実施形態は、暗号文にf_1、f_2、・・・、f_nを含めることで、ｎ人の相手に対してマルチキャストを行い、そのｎ人のうちの何人かは単独で復号可能とし、残りの者は少なくとも二人が協力することで、復号可能とすることもできる。あるいは、ｎ人各々が単独で復号可能とすることもできる。

【０４１６】図２４は、本発明の第五実施形態を構成するデータ暗号化装置のしきい値ロジック部の変形例の機能構成を示す図である。

【０４１７】図２４に示すように、しきい値ロジック部2125ａには、３つのデータ$x_1$2402、$x_2$2403、$x_3$2404が入力される。

【０４１８】ここで、$x_1$2402、$x_2$2403、$x_3$2406は、それぞれ、図２１に示す整数倍演算部２１１４、２１１５、２１１６で生成されたｘ座標値である。

【０４１９】演算部２４０８は、次式で示される処理を実行することで、データ$f_1$2406を生成する。

【０４２０】$f_1 = x_1 - h(x_2)$

ここで、ｈ（ｘ2）は、図２１に示すハッシュ値生成装置２１２６で生成されたハッシュ値である。

【０４２１】また、演算部２４１０は、次式で示される処理を実行することで、データ$f_2$2407を生成する。

【０４２２】$f_2 = x_1 - h(x_3)$

ここで、ｈ（ｘ3）は、図２１に示すハッシュ値生成装置２１２７で生成されたハッシュ値である。

【０４２３】図２４に示すしきい値ロジック部２１２５ａを用いることで、図２１に示すデータ暗号化装置２１０１は、暗号化処理途中に生成される乱数ｋと公開鍵$Q_2$２１０４とを基にデータ$f_1$2406を生成することになる。このため、公開鍵$Q_2$２１０４と対の関係にある秘密鍵d_2の持ち主も、暗号文２１０８を単独で復号することができる。

【０４２４】同様に、暗号化処理途中に生成される乱数ｋと公開鍵$Q_3$２１０５とを基に、データ$f_2$2407を生成することになる。このため、公開鍵$Q_3$２１０５と対の関係にある秘密鍵d_3の持ち主も、暗号文２１０８を単独で復号することができる。

【０４２５】すなわち、暗号文２１０８を秘密鍵d_1、d_2、d_3のそれぞれの持ち主に同報通信（マルチキャスト）することで、これ等の持ち主各々が単独で復号することが可能となる。また、通信相手を一人増やすには、単に、ハッシュ値の長さ分（たとえば、８０ビット）のデータを増やせばよいので、同報暗号通信を効率的に行うことができる。ここで、ハッシュ関数には一方向性だけが要求され、衝突回避性は要求されない。

【０４２６】なお、上記の実施形態では、楕円曲線暗号として、

$y^2 = x^3 + ax + b$

に基づくものを用いたが、その代わりに、楕円曲線暗号として、$y^2 + xy = x^3 + ax + b$に基づくものを用いてもよい。

【０４２７】

【発明の効果】以上説明したように本発明によれば、データ攪乱度の高いハッシュ値を、迅速に生成することができる。

【０４２８】また、データ攪乱度の高い鍵や暗号文などのデータを、迅速に生成することができる。

【０４２９】さらに、同じデータを複数の宛先に暗号化して送信しようとする場合に、各宛先毎に、当該宛先から予め配布されて公開鍵を用いて暗号化処理を行わなう必要がなくなる。

【０４３０】また、間違ってファイルから復号鍵を消去するなどして、復号鍵を紛失してしまった場合でも、自己宛てに送られてきた暗号化データを、他の二人以上の受信者が協力することで復号することが可能になる。

【図面の簡単な説明】

【図１】本発明の第一実施形態であるハッシュ値生成装

置の機能構成を示す図である。

【図２】図１に示す混合処理部１０３での処理の一例を説明するための図である。

【図３】図に示す伸長処理部１０４での処理の一例を説明するためのフロー図である。

【図４】図１に示す単射拡大部１０５での処理の一例を説明するためのフロー図である。

【図５】図４に示すステップ４０５における６４ビットデータから９６ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【図６】図４に示すステップ４０６における９６ビットデータから１２８ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【図７】図４に示すステップ４０７における１２８ビットデータから２５６ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【図８】本発明の第一実施形態の変形例であるハッシュ値生成装置の機能構成を示す図である。

【図９】図８に示す混合処理部８０１での処理の一例を説明するための図である。

【図１０】図８に示す単射拡大部８０３での処理の一例を説明するためのフロー図である。

【図１１】図１０に示すステップ１００５における６４ビットデータから６４ビットデータへの単射処理の一例を説明するためのフロー図である。

【図１２】図１０に示すステップ１００６における６４ビットデータから８０ビットデータへの単射拡大処理の一例を説明するためのフロー図である。

【図１３】本発明の第二実施形態であるデータ暗号化装置の機能構成を示す図である。

【図１４】図１３に示すπ関数処理部１３０６～１３１３における処理の一例を説明するためのフロー図である。

【図１５】本発明の第三実施形態であるマスキング装置の機能構成を示した図である。

【図１６】本発明の第四実施形態であるデータ暗号化システムを構成するデータ暗号化装置の機能構成を示す図である。

【図１７】図１６に示す圧縮・暗号化部１６１２、１６１５、・・・の機能構成を示す図である。

【図１８】本発明の第四実施形態であるデータ暗号化システムを構成するデータ復号化装置の機能構成を示す図である。

【図１９】図１８に示す復号・伸長化部１８１０、１８１２、・・・の機能構成を示す図である。

【図２０】図１６に示す本発明の第四実施形態におけるデータ暗号化装置の変形例の機能構成を示す図である。

【図２１】本発明の第五実施形態であるデータ暗号化システムを構成するデータ暗号化装置の機能構成を示す図である。

【図２２】図２１に示すしきい値ロジック部２１２５の機能構成を示す図である。

【図２３】図２１に示す圧縮・暗号化部２１２０、・・・の機能構成を示す図である。

【図２４】本発明の第五実施形態を構成するデータ暗号化装置のしきい値ロジックの変形例の機能構成を示す図である。

【図２５】本発明の第五実施形態を構成するデータ暗号化装置の変形例の機能構成を示す図である。

【図２６】本発明の第五実施形態であるデータ暗号化システムを構成するデータ復号化装置の機能構成を示す図である。

【図２７】従来のハッシュ関数を説明するための図である。

【符号の説明】

１０１、１０１ａ、１３０１、１５０９、１８０９、１９１２、２０１１、２０１４、２１１９、２１２１、２１２６～２１２８、２３２３、２６２０、２６２３、２６１５　ハッシュ値生成装置

１０２　データ拡大部

１０３、８０１　混合処理部

１０４　伸長処理部

１０５、８０３　単射拡大部

１３１１、１６０１、１８０１、２００１　データ暗号化装置

１３０６～１３１３、１５１３、１５１４、１５１８、１５１９、１７１０、１７１１、１７１６、１７１８、２３１５、２３１６、２３２２、２３２４　π関数処理部

１５０１　マスキング生成装置

１５０８、１６０７、２００７、２１１３　乱数生成装置

１６０８、１６０９、１８０７、２００８、２００９、２１１４～２１１６、２１２３、２１２４、２５０１、２６１９、２６２２　整数倍演算部

１６１０、１６１３、１８０８、１８１１　番号生成部

１６１２、１６１５、２０１２、２０１５、２１２０　圧縮・暗号化部

１７０７、１７１３、２３１２、２３１９　圧縮処理部

１７０６、１７１２、１７１７、１９０６、１９０９、１９１５、２３１０、２３１７　拡張部

１８１０、１８１３　復号伸長部

１９１１、１９１７、２６１４、２６１７　伸長処理部

１９０７、１９１０、１９１４、１９１６　π^{-1}関数処理部

２１２５　しきい値ロジック部

２３２５、２６１１　演算部

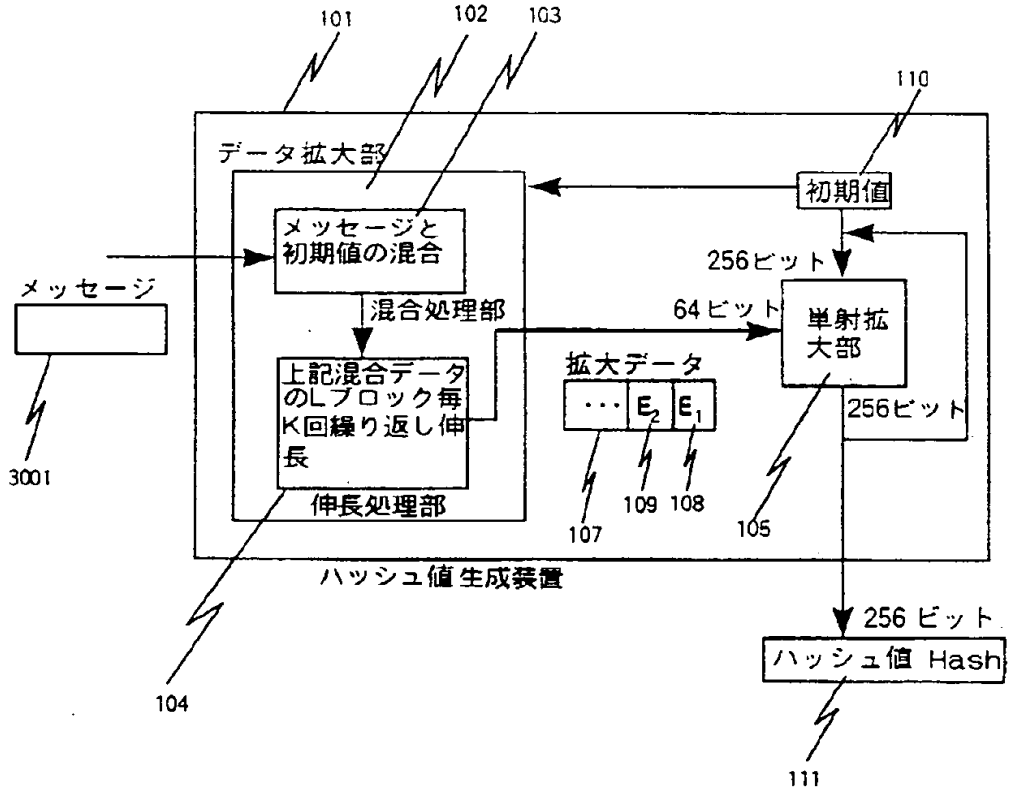
２６０７　データ回復機関

２６０８、２６０９　鍵保管機関

２６１２　しきい値逆ロジック部

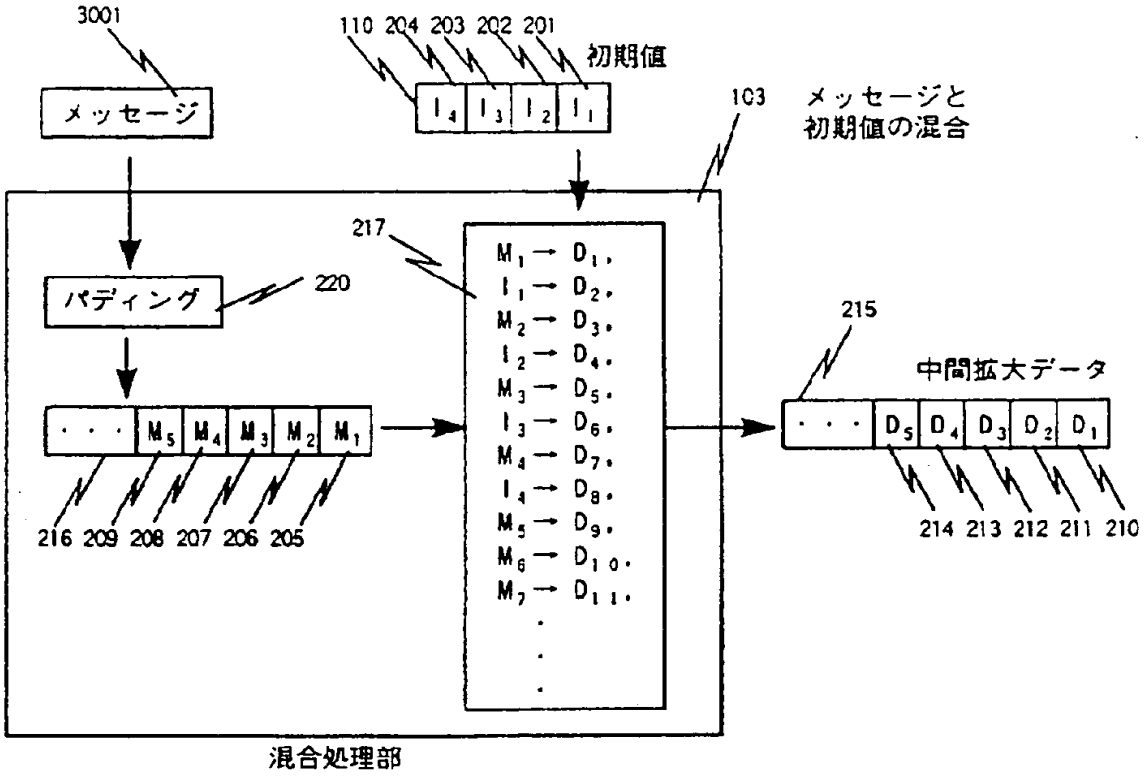
【図１】

図１


101
102
103
110
データ拡大部
初期値
メッセージと
初期値の混合
256ビット
メッセージ
混合処理部
64ビット
単射拡
大部
拡大データ
上記混合データ
のLブロック毎
K回繰り返し伸
長
・・・ E2 E1
256ビット
3001
伸長処理部
107
109
108
105
ハッシュ値生成装置
256 ビット
ハッシュ値 Hash
104
111

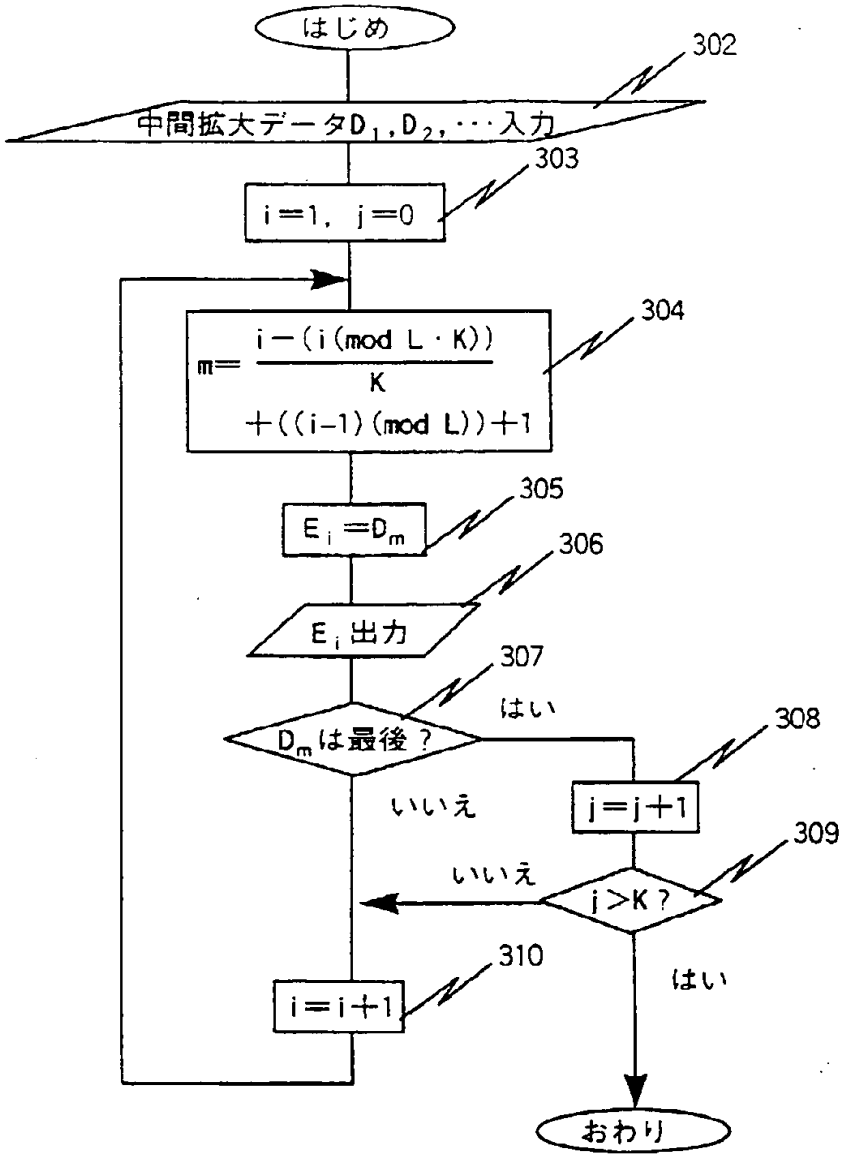
【図２】

図2


3001
メッセージ
110 204 203 202 201
初期値
I4 I3 I2 I1
103
メッセージと
初期値の混合
217
M1 → D1.
I1 → D2.
M2 → D3.
I2 → D4.
M3 → D5.
I3 → D6.
M4 → D7.
I4 → D8.
M5 → D9.
M6 → D10.
M7 → D11.
パディング
220
・・・ M5 M4 M3 M2 M1
216 209 208 207 206 205
215
中間拡大データ
・・・ D5 D4 D3 D2 D1
214 213 212 211 210
混合処理部

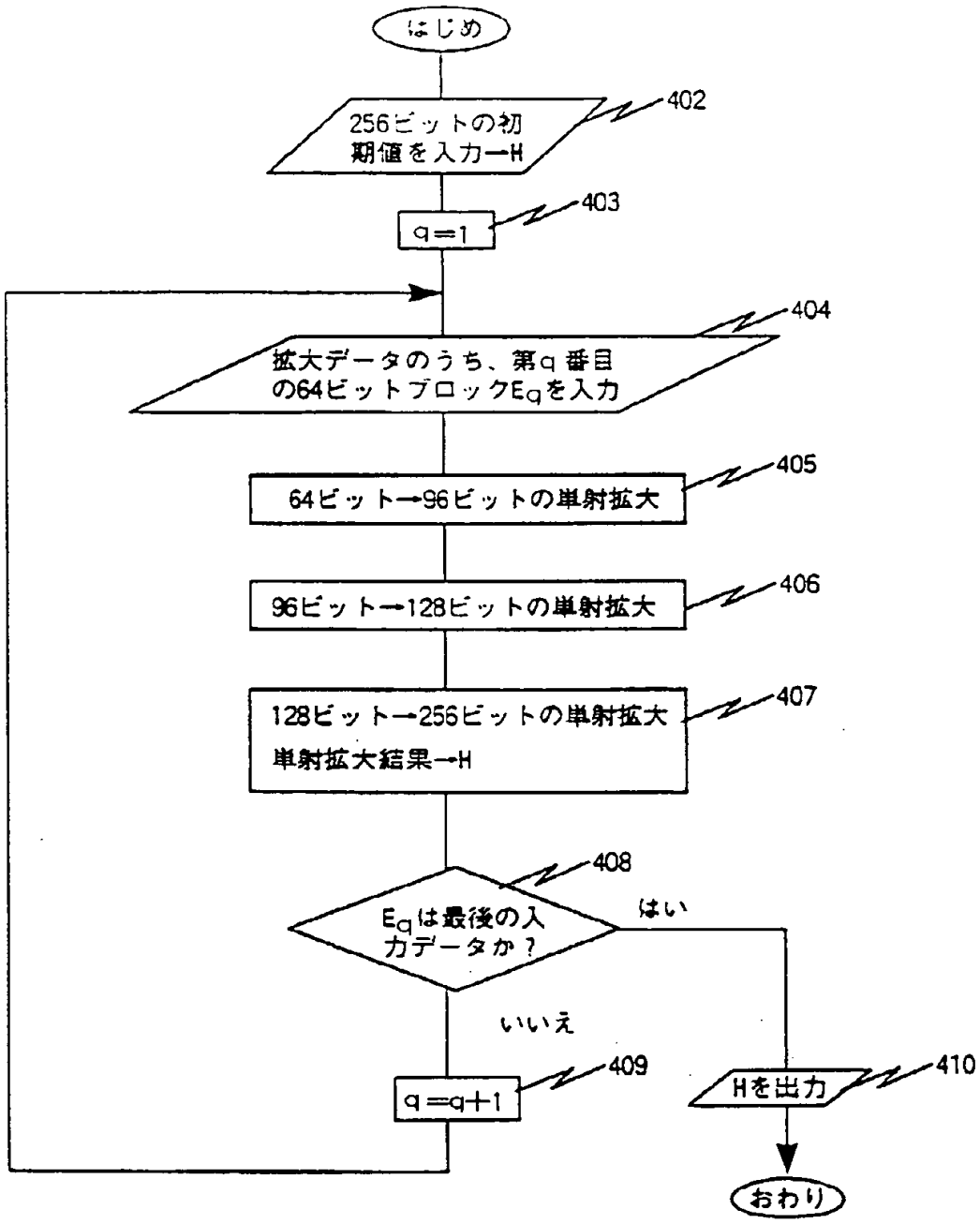
【図３】

図３


はじめ
302
中間拡大データ$D_1, D_2, \cdots$入力
303
i＝1，j＝0
304
$m=\frac{i-(i(\mathrm{mod}\ L\cdot K))}{K}+((i-1)(\mathrm{mod}\ L))+1$
305
$E_i=D_m$
306
E_i出力
307
D_mは最後？
はい
308
j＝j＋1
いいえ
309
j＞K？
いいえ
はい
310
i＝i＋1
おわり

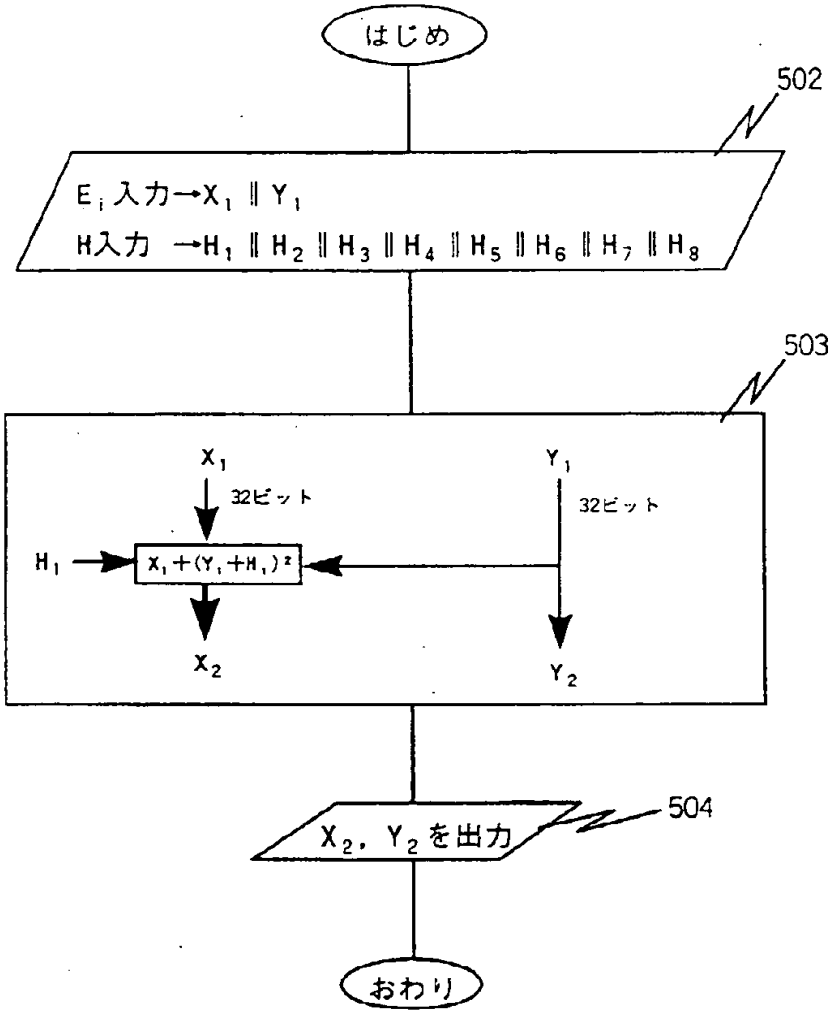
【図４】

図4


はじめ
256ビットの初期値を入力→H
402
q=1
403
拡大データのうち、第q番目の64ビットブロックEqを入力
404
64ビット→96ビットの単射拡大
405
96ビット→128ビットの単射拡大
406
128ビット→256ビットの単射拡大
単射拡大結果→H
407
Eqは最後の入力データか？
408
はい
いいえ
q=q+1
409
Hを出力
410
おわり

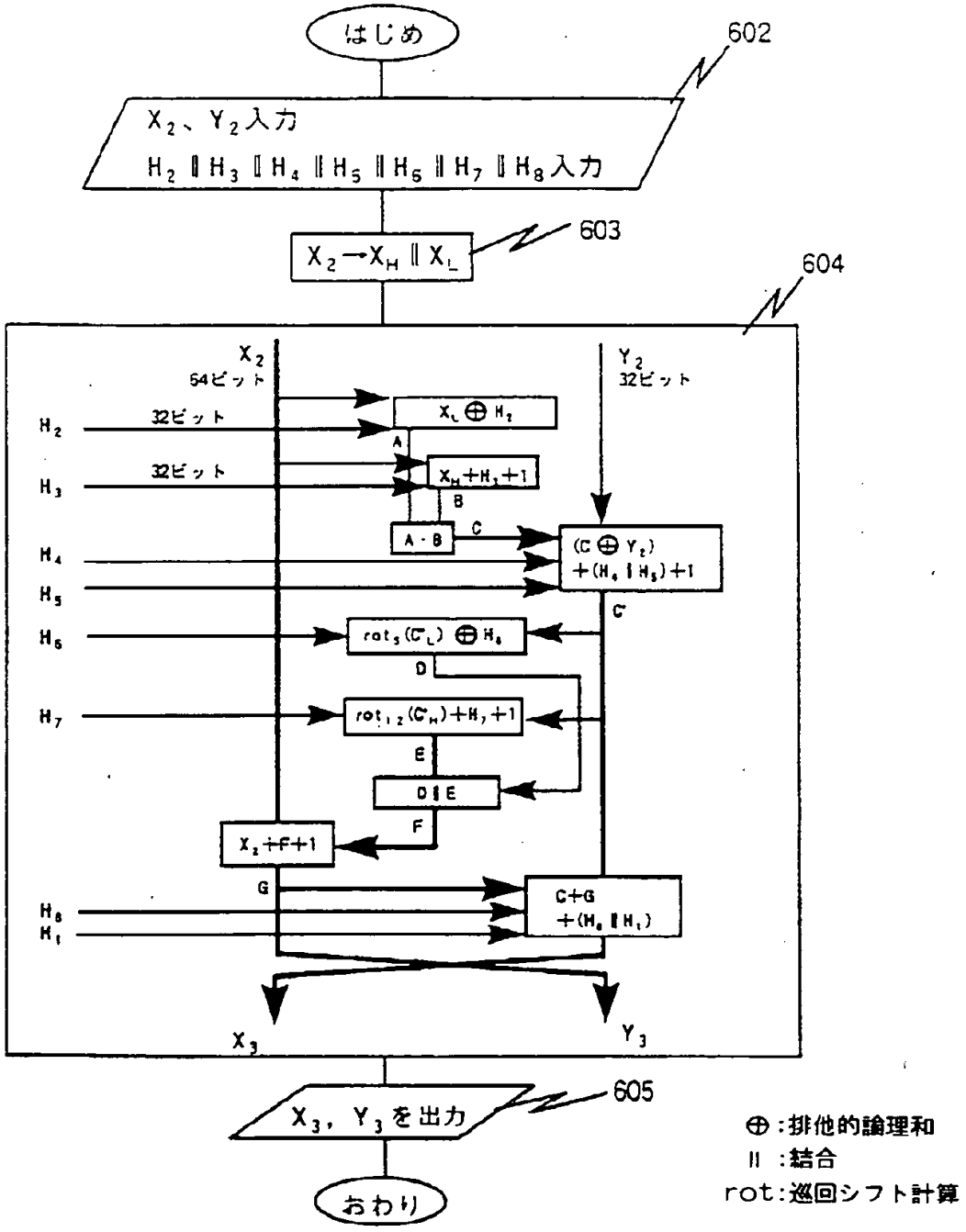
【図５】

図５


はじめ
502
E_i入力→$X_1 \| Y_1$
H入力 →$H_1 \| H_2 \| H_3 \| H_4 \| H_5 \| H_6 \| H_7 \| H_8$
503
X_1
32ビット
Y_1
32ビット
H_1
$X_1+(Y_1+H_1)^2$
X_2
Y_2
504
X_2，Y_2を出力
おわり

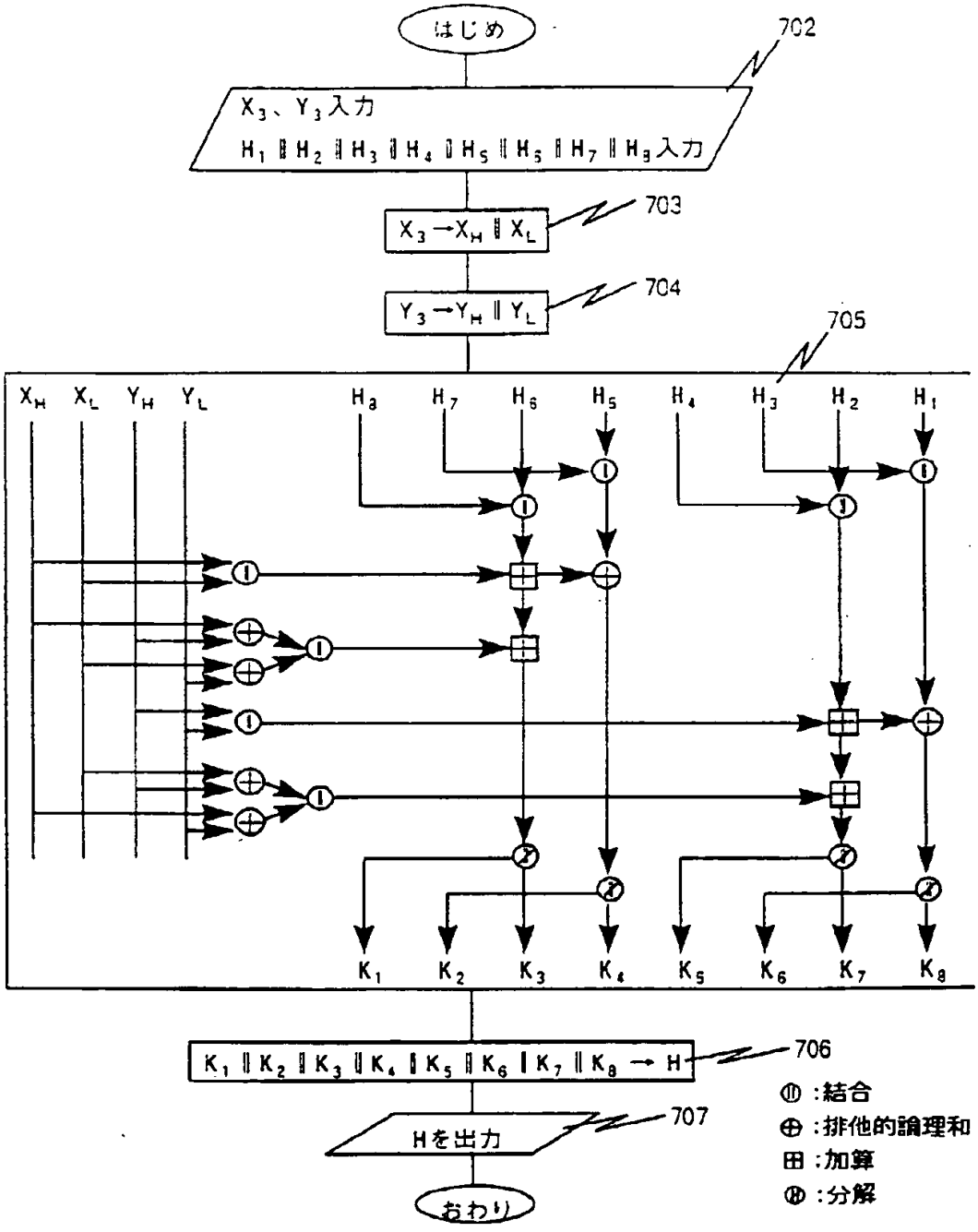
【図６】

図6


はじめ
602
X2、Y2入力
H2 ‖ H3 ‖ H4 ‖ H5 ‖ H6 ‖ H7 ‖ H8入力
603
X2→XH ‖ XL
604
X2
64ビット
Y2
32ビット
H2
32ビット
H3
32ビット
XL ⊕ H2
A
XH+H3+1
B
A・B
C
(C ⊕ Y2)
+(H4 ‖ H5)+1
H4
H5
C'
H6
rot3(C'L) ⊕ H6
D
H7
rot12(C'H)+H7+1
E
D ‖ E
F
X2+F+1
G
C+G
+(H8 ‖ H1)
H8
H1
X3
Y3
605
X3, Y3を出力
おわり
⊕:排他的論理和
‖ :結合
rot:巡回シフト計算

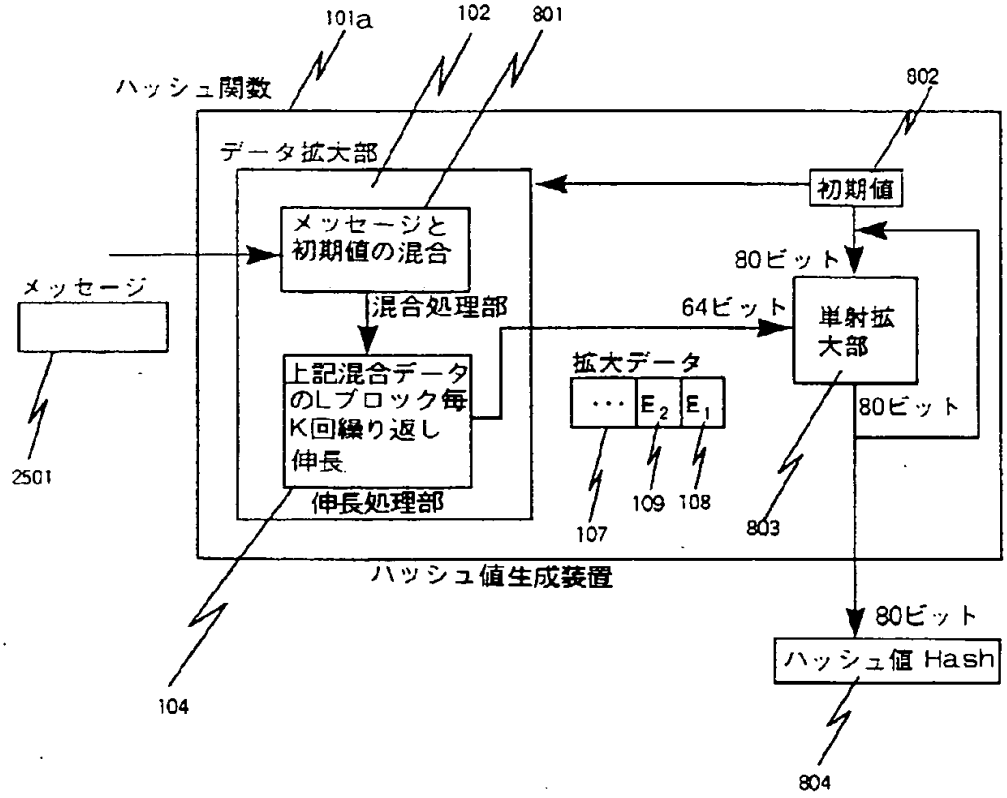
【図７】

図7


はじめ
702
X3、Y3入力
H1‖H2‖H3‖H4‖H5‖H6‖H7‖H8入力
X3→XH‖XL
703
Y3→YH‖YL
704
705
XH XL YH YL
H8 H7 H6 H5 H4 H3 H2 H1
K1 K2 K3 K4 K5 K6 K7 K8
K1‖K2‖K3‖K4‖K5‖K6‖K7‖K8 → H
706
Hを出力
707
おわり
⑪ :結合
⊕ :排他的論理和
⊞ :加算
⑫ :分解

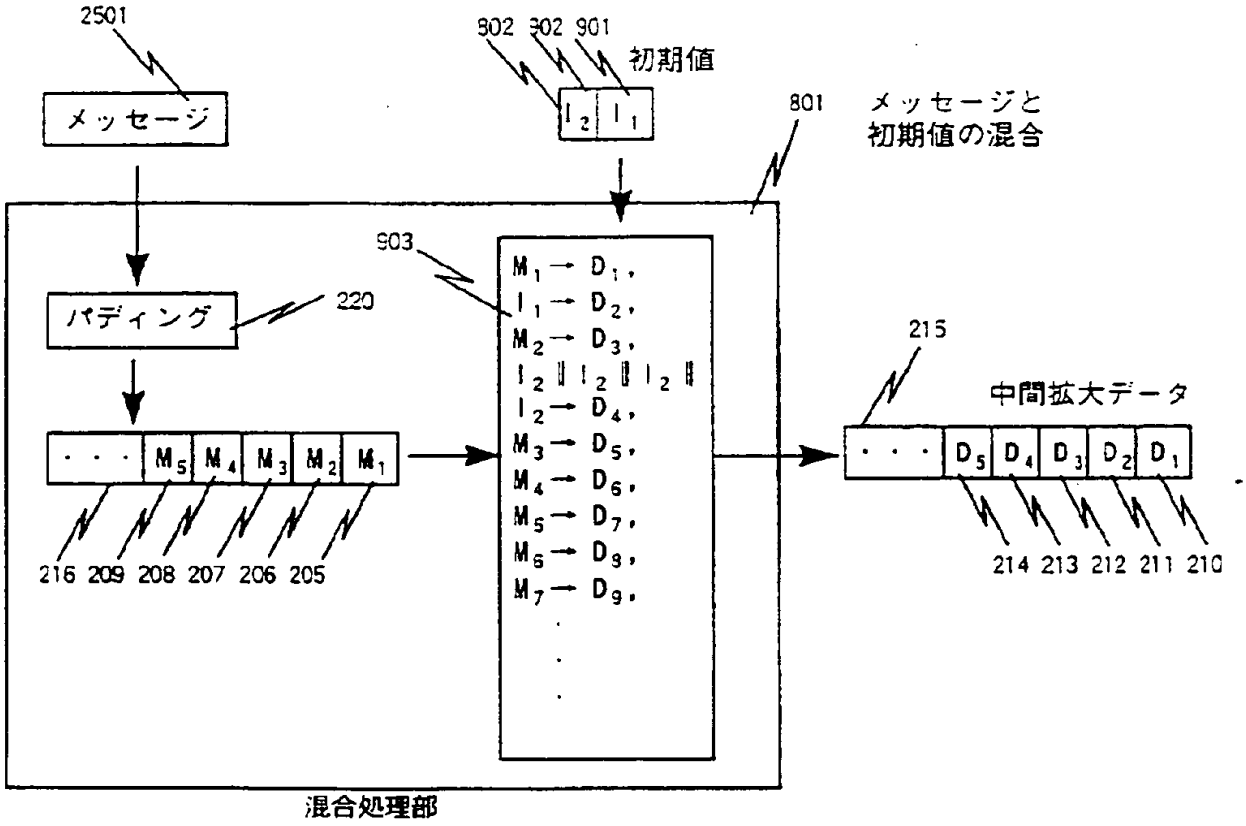
【図８】

図８


101a
102
801
ハッシュ関数
802
データ拡大部
初期値
メッセージと
初期値の混合
80ビット
メッセージ
混合処理部
64ビット
単射拡
大部
上記混合データ
のLブロック毎
K回繰り返し
伸長
拡大データ
・・・ E2 E1
80ビット
2501
伸長処理部
107
109
108
803
ハッシュ値生成装置
80ビット
ハッシュ値 Hash
104
804

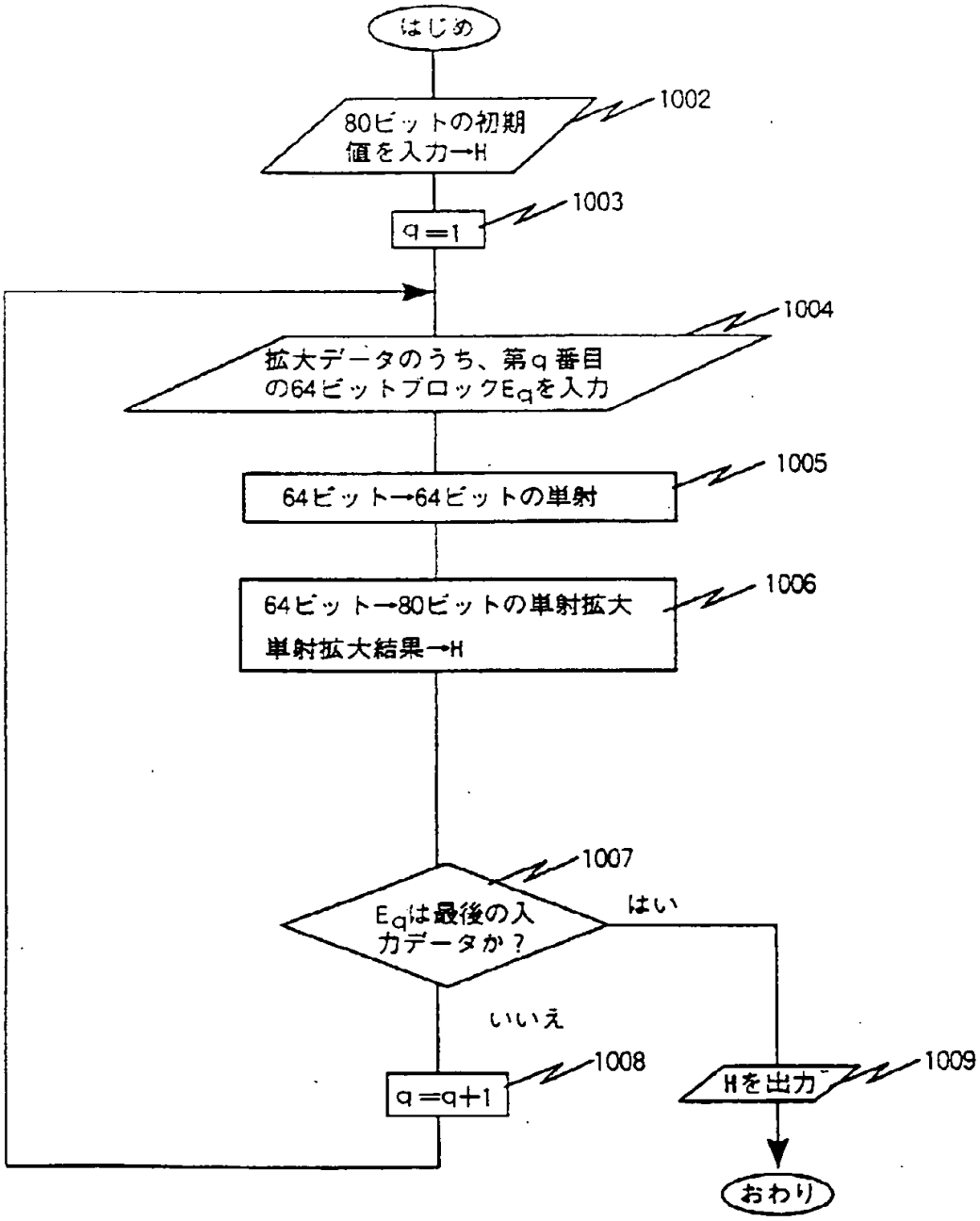
【図９】

図9


2501
メッセージ
902 902 901
初期値
I_2 I_1
801
メッセージと
初期値の混合
903
パディング
220
$M_1 \rightarrow D_1$,
$I_1 \rightarrow D_2$,
$M_2 \rightarrow D_3$,
I_2 I_2 I_2
$I_2 \rightarrow D_4$,
$M_3 \rightarrow D_5$,
$M_4 \rightarrow D_6$,
$M_5 \rightarrow D_7$,
$M_6 \rightarrow D_8$,
$M_7 \rightarrow D_9$,
215
中間拡大データ
・・・ M_5 M_4 M_3 M_2 M_1
・・・ D_5 D_4 D_3 D_2 D_1
216 209 208 207 206 205
214 213 212 211 210
混合処理部

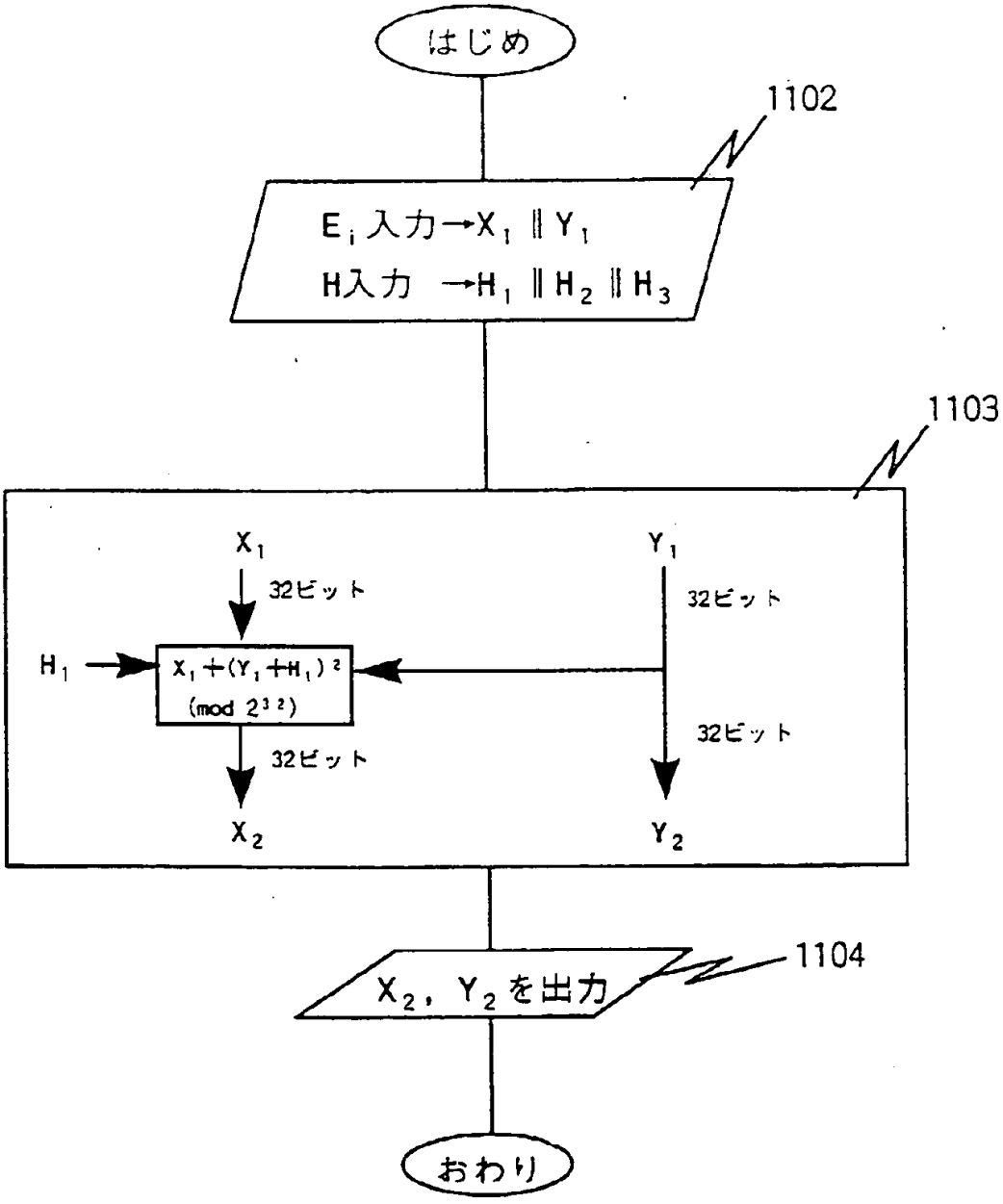
【図１０】

図10


はじめ
80ビットの初期値を入力→H
1002
q=1
1003
拡大データのうち、第q番目の64ビットブロックEqを入力
1004
64ビット→64ビットの単射
1005
64ビット→80ビットの単射拡大
単射拡大結果→H
1006
Eqは最後の入力データか？
1007
はい
いいえ
q=q+1
1008
Hを出力
1009
おわり

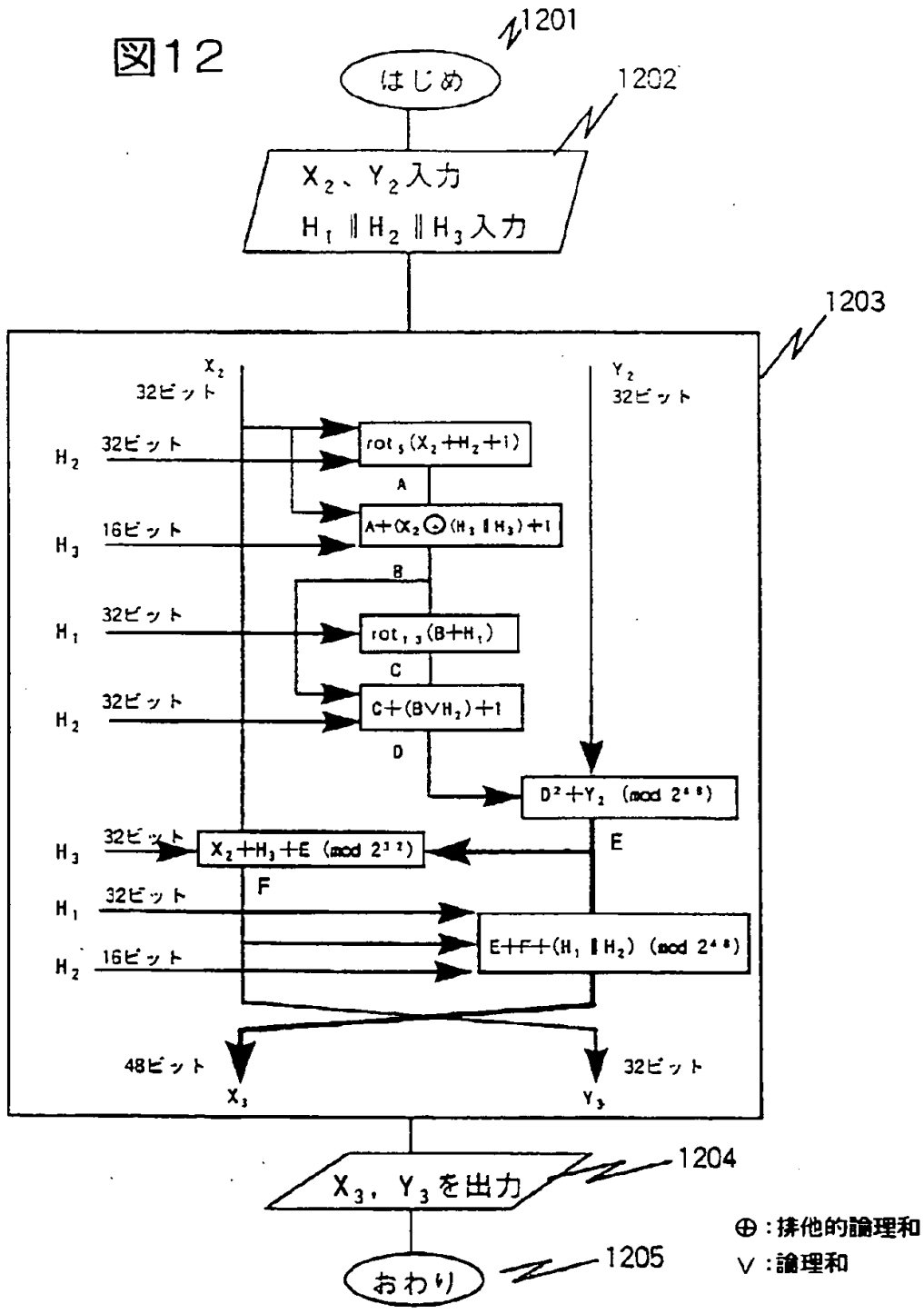
【図１１】

図11


はじめ
1102
E_i入力→$X_1 \| Y_1$
H入力 →$H_1 \| H_2 \| H_3$
1103
X_1
32ビット
Y_1
32ビット
H_1
$X_1+(Y_1+H_1)^2$
(mod 2^{32})
32ビット
32ビット
X_2
Y_2
1104
X_2, Y_2を出力
おわり

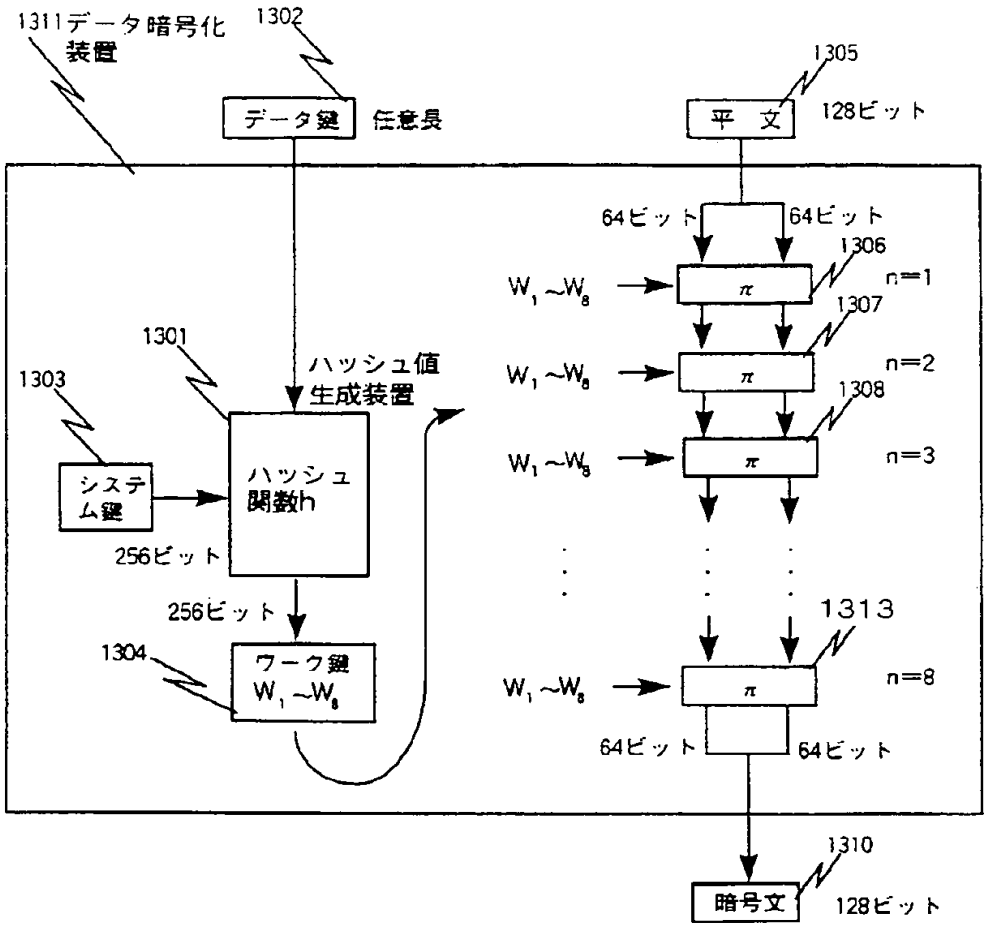
【図１２】

図12


1201
はじめ
1202
X2、Y2入力
H1 ‖ H2 ‖ H3入力
1203
X2
32ビット
Y2
32ビット
H2
32ビット
rot5(X2+H2+1)
A
H3
16ビット
A+(X2⊙(H3 ‖ H3))+1
B
H1
32ビット
rot1,3(B+H1)
C
H2
32ビット
C+(B∨H2)+1
D
D²+Y2 (mod 2⁴⁸)
E
H3
32ビット
X2+H3+E (mod 2³²)
F
H1
32ビット
H2
16ビット
E+F+(H1 ‖ H2) (mod 2⁴⁸)
48ビット
32ビット
X3
Y3
1204
X3, Y3を出力
1205
おわり
⊕：排他的論理和
∨：論理和

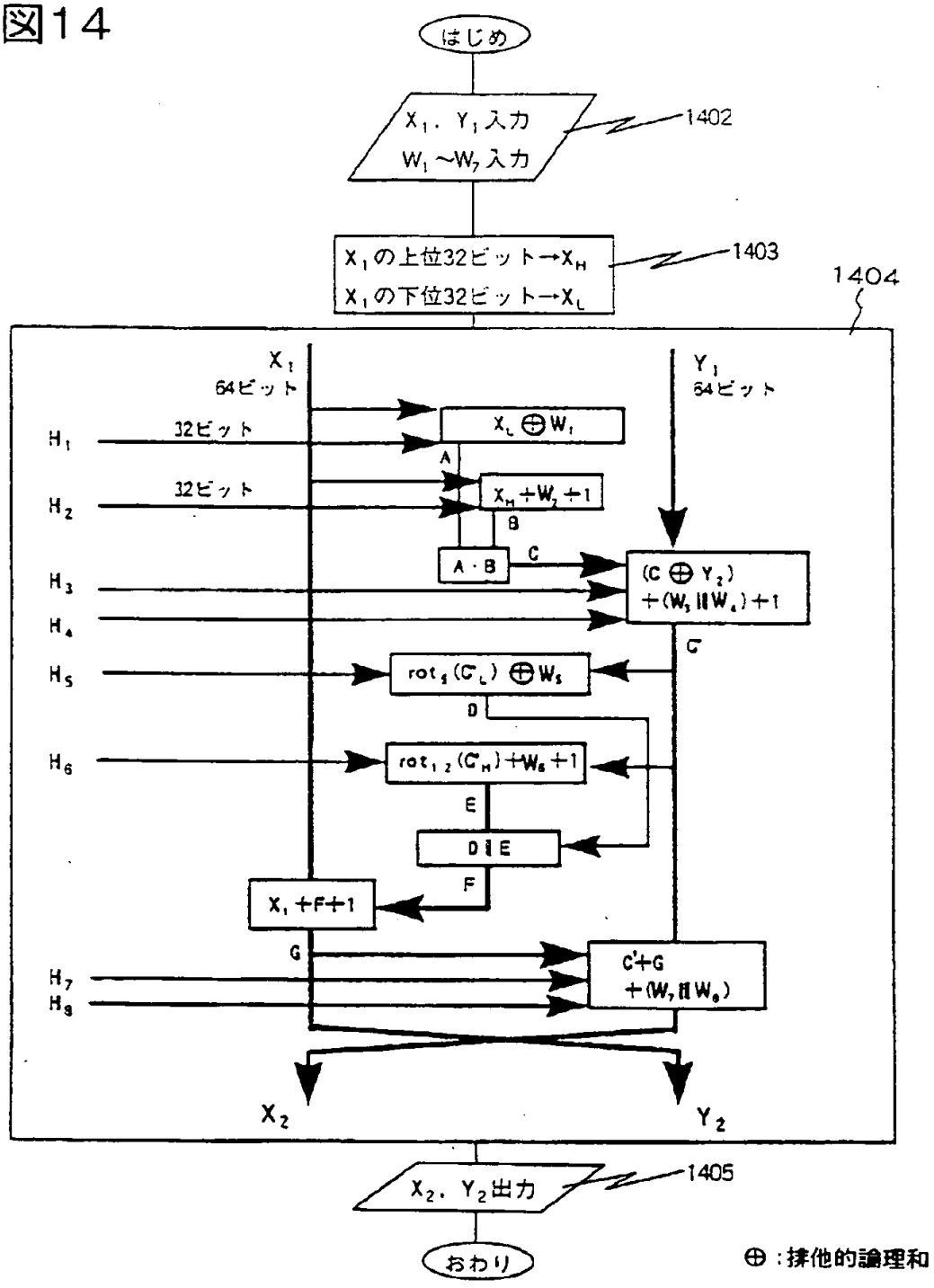
【図１３】

図１３


1311データ暗号化装置
1302
データ鍵
任意長
1305
平　文
128ビット
64ビット
64ビット
1306
n=1
1307
n=2
1308
n=3
1313
n=8
1301
ハッシュ値生成装置
1303
システム鍵
ハッシュ関数h
256ビット
256ビット
1304
ワーク鍵
W1～W8
64ビット
64ビット
1310
暗号文
128ビット

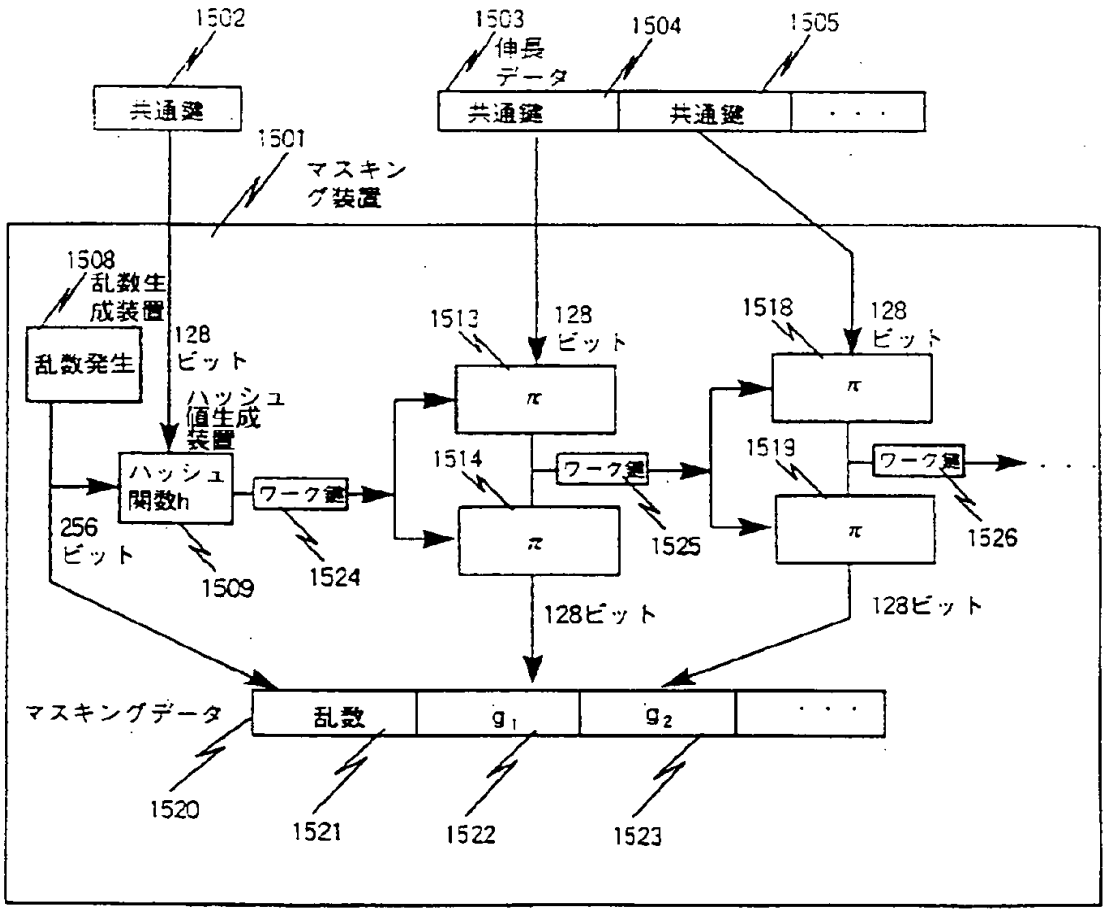
【図１４】

図14


はじめ
X1．Y1入力
W1～W7入力
1402
X1の上位32ビット→XH
X1の下位32ビット→XL
1403
1404
X1
64ビット
Y1
64ビット
H1
32ビット
H2
32ビット
XL⊕W1
A
XH+W2+1
B
A・B
C
(C⊕Y2)
+(W3||W4)+1
C'
H3
H4
H5
rot5(C'L)⊕W5
D
H6
rot12(C'H)+W6+1
E
D||E
F
X1+F+1
G
C'+G
+(W7||W8)
H7
H8
X2
Y2
X2，Y2出力
1405
おわり
⊕：排他的論理和

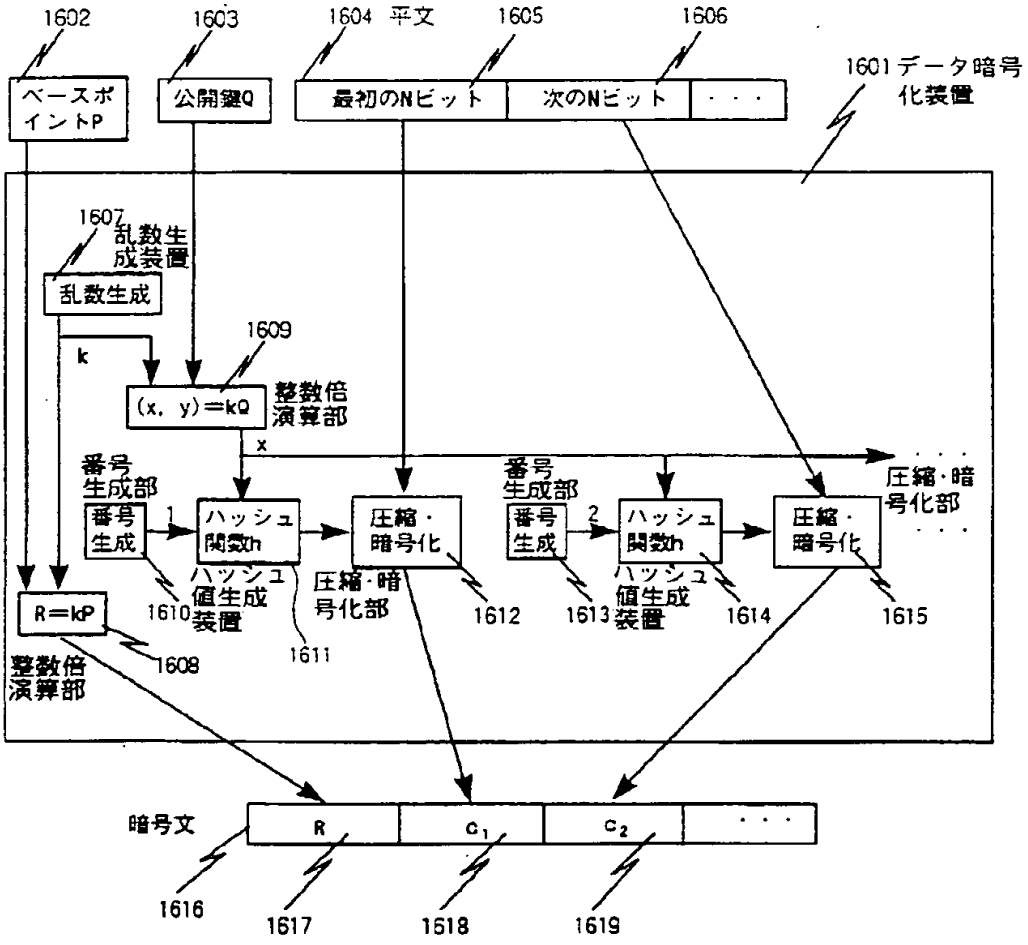
【図１５】

図15


1502
共通鍵
1503
伸長
データ
1504
1505
共通鍵
共通鍵
・・・
1501
マスキング装置
1508
乱数生成装置
乱数発生
128
ビット
ハッシュ値生成装置
ハッシュ関数h
256
ビット
1509
ワーク鍵
1524
1513
128
ビット
π
1514
ワーク鍵
1525
π
128ビット
1518
128
ビット
π
1519
ワーク鍵
1526
π
128ビット
マスキングデータ
乱数
g1
g2
・・・
1520
1521
1522
1523

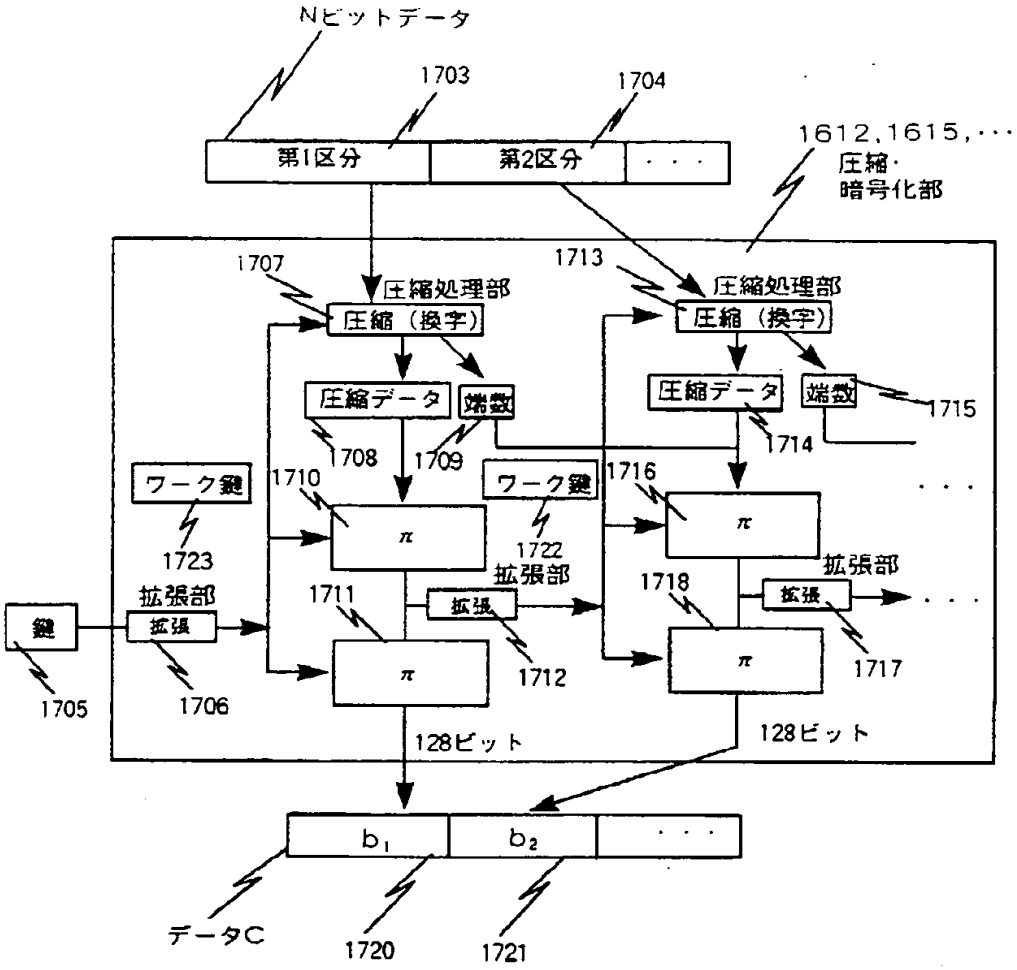
【図１６】

図１６


1602
1603
1604 平文
1605
1606
1601 データ暗号化装置
ベースポイントP
公開鍵Q
最初のNビット
次のNビット
・・・
1607 乱数生成装置
乱数生成
k
1609
(x, y)＝kQ
整数倍演算部
x
番号生成部
番号生成
1
ハッシュ関数h
圧縮・暗号化
番号生成部
番号生成
2
ハッシュ関数h
圧縮・暗号化
圧縮・暗号化部
R＝kP
整数倍演算部
1608
1610 ハッシュ値生成装置
1611 圧縮・暗号化部
1612
1613 ハッシュ値生成装置
1614
1615
暗号文
R
C1
C2
・・・
1616
1617
1618
1619

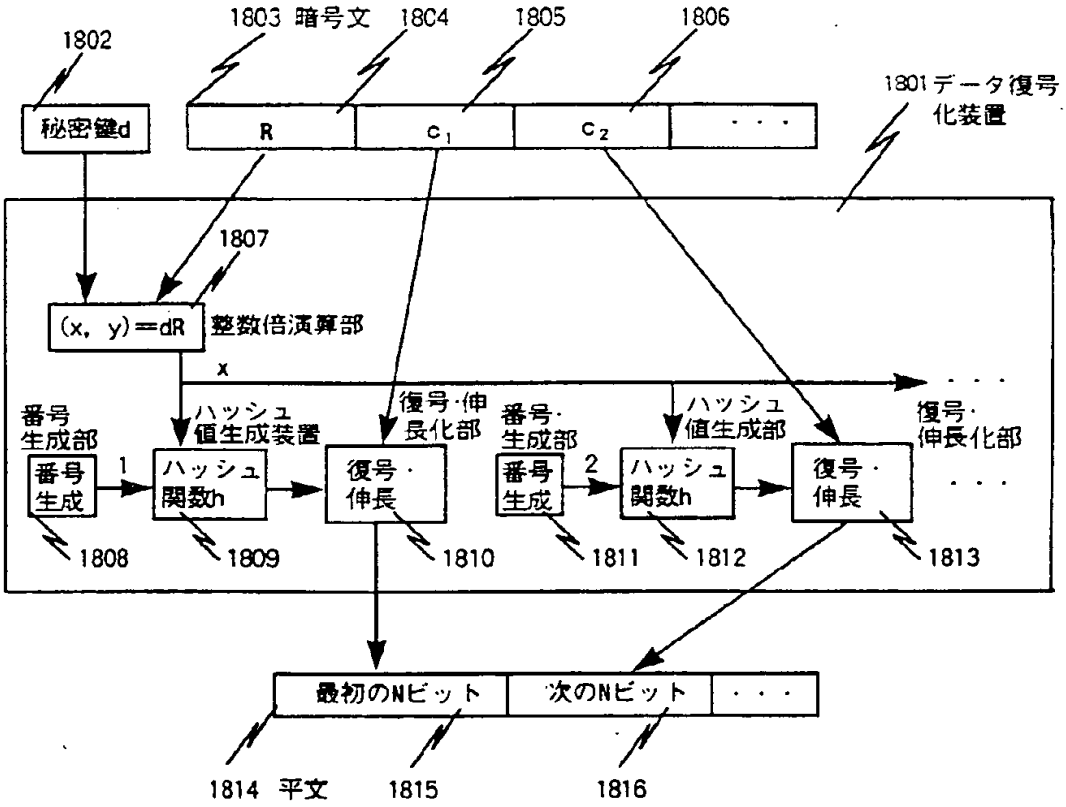
【図１７】

図１７


Nビットデータ
1703
1704
第1区分
第2区分
・・・
1612,1615,・・・
圧縮・
暗号化部
1707
圧縮処理部
圧縮（換字）
1713
圧縮処理部
圧縮（換字）
圧縮データ
端数
圧縮データ
端数
1715
1708
1709
1714
ワーク鍵
1710
ワーク鍵
1716
π
π
1723
1722
拡張部
拡張部
1718
鍵
拡張
1711
拡張
拡張
1705
1706
π
1712
π
1717
128ビット
128ビット
b1
b2
・・・
データC
1720
1721

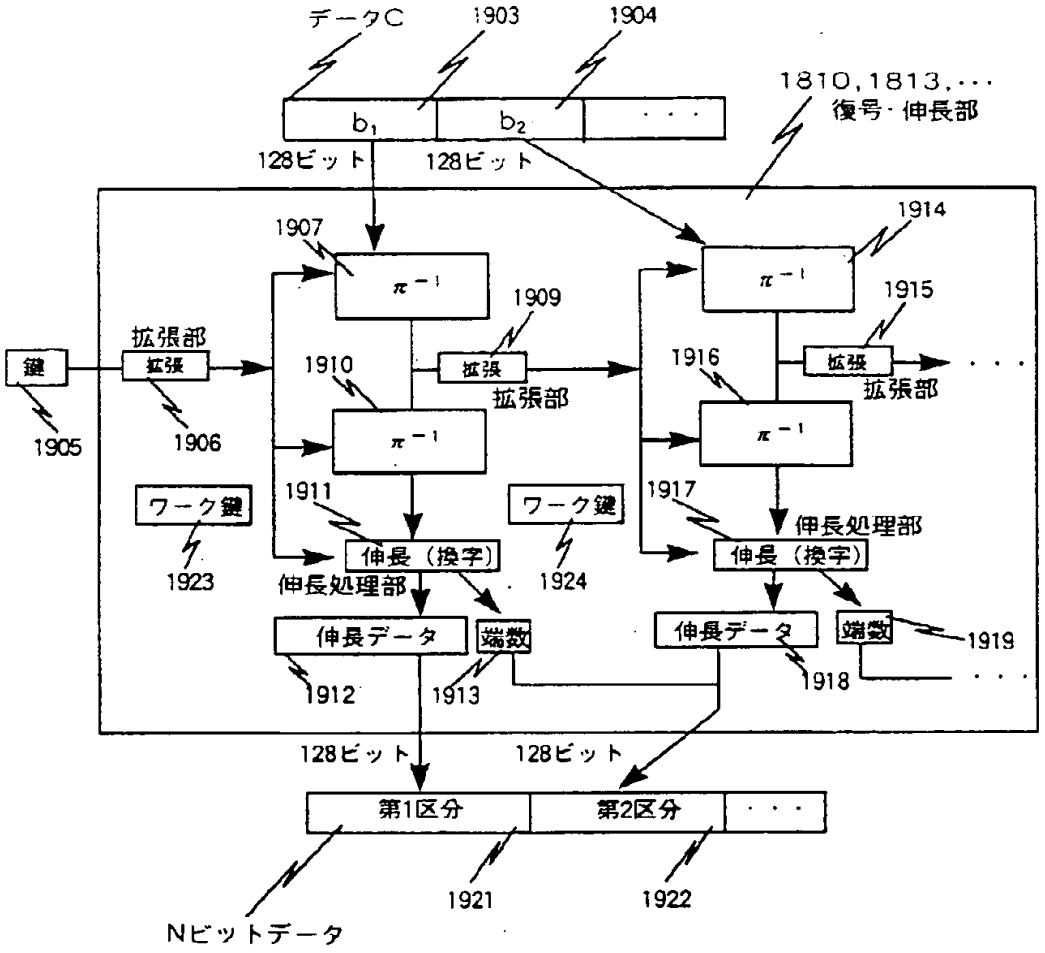
【図１８】

図１８


1802
1803 暗号文
1804
1805
1806
1801データ復号化装置
秘密鍵d
R
c1
c2
・・・
1807
(x, y)＝dR
整数倍演算部
x
・・・
番号生成部
ハッシュ値生成装置
復号・伸長化部
番号・生成部
ハッシュ値生成部
復号・伸長化部
番号生成
1
ハッシュ関数h
復号・伸長
番号生成
2
ハッシュ関数h
復号・伸長
・・・
1808
1809
1810
1811
1812
1813
最初のNビット
次のNビット
・・・
1814 平文
1815
1816

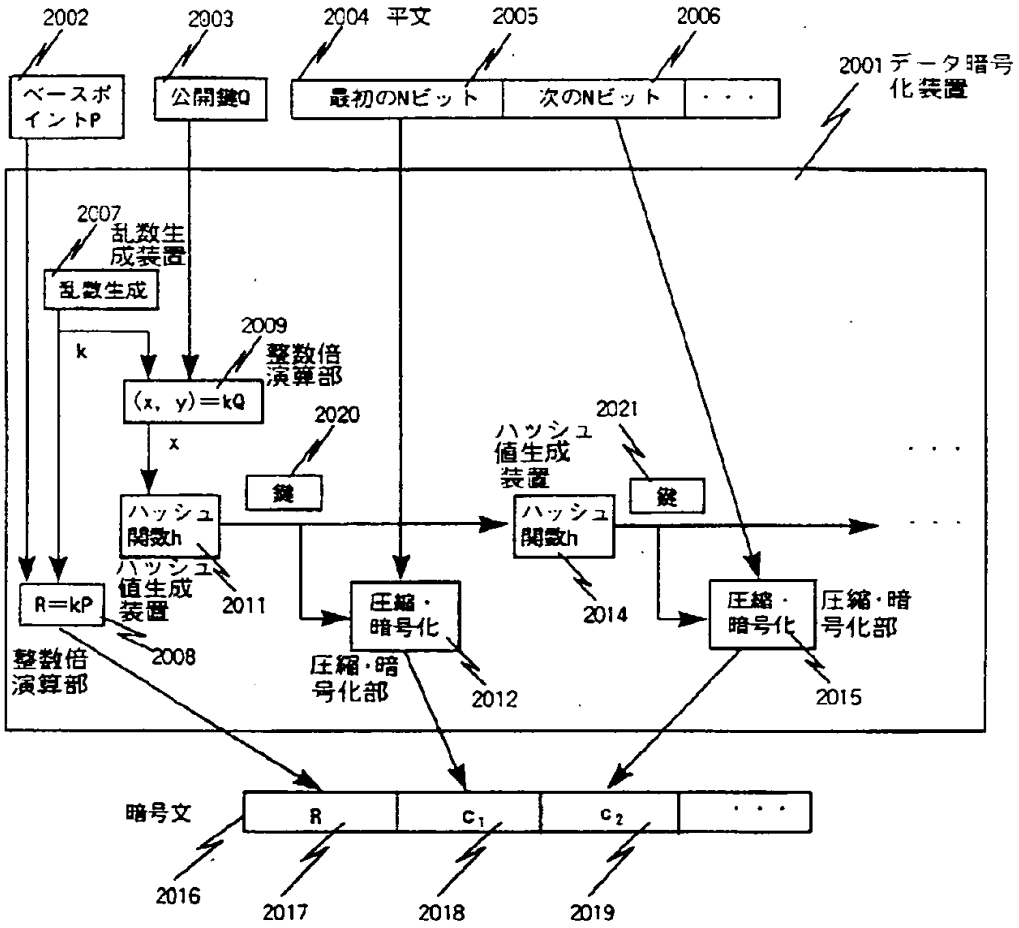
【図１９】

図１９


データC
1903
1904
1810,1813,・・・
復号・伸長部
b1
b2
・・・
128ビット
128ビット
1907
1914
π⁻¹
π⁻¹
1909
1915
鍵
拡張部
拡張
拡張
拡張部
拡張部
1905
1906
1910
1916
π⁻¹
π⁻¹
ワーク鍵
1911
ワーク鍵
1917
伸長処理部
伸長（換字）
伸長（換字）
1923
伸長処理部
1924
伸長データ
端数
伸長データ
端数
1919
1912
1913
1918
128ビット
128ビット
第1区分
第2区分
・・・
1921
1922
Nビットデータ

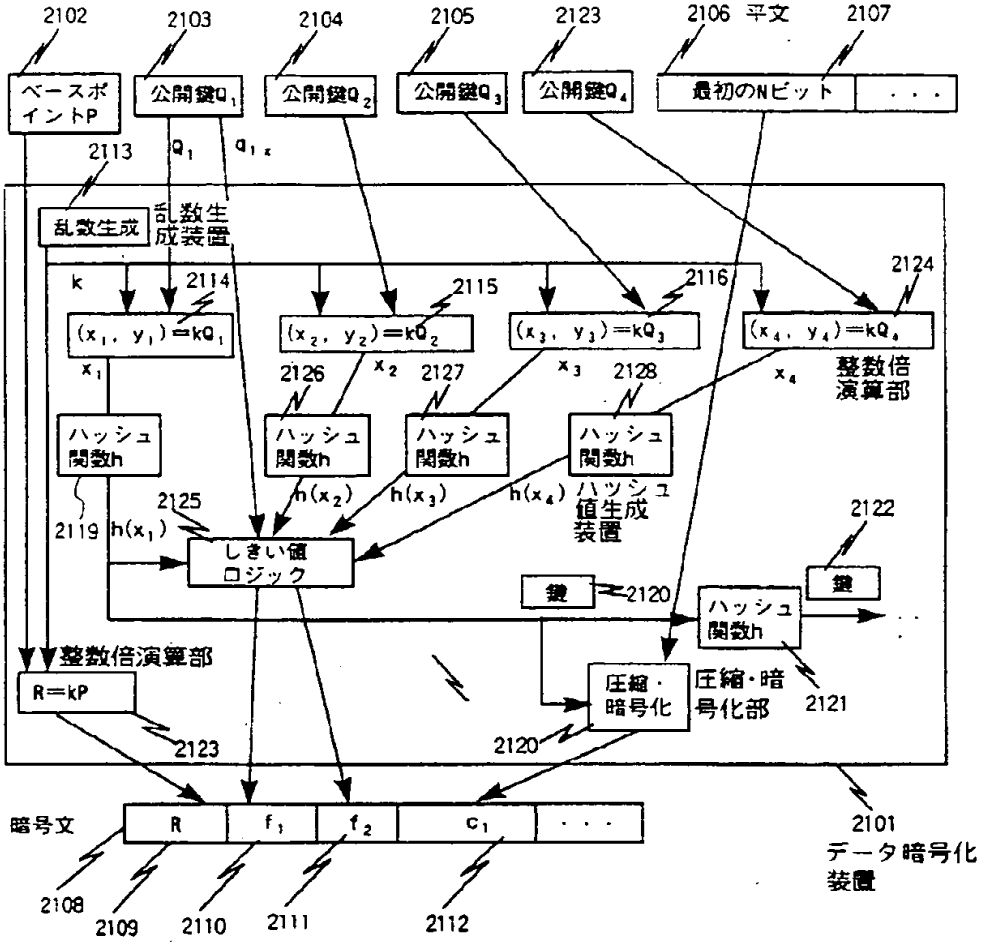
【図２０】

図２０


2002
2003
2004 平文
2005
2006
2001 データ暗号化装置
ベースポイントP
公開鍵Q
最初のNビット
次のNビット
・・・
2007
乱数生成装置
乱数生成
k
2009
整数倍演算部
(x, y)=kQ
x
2020
鍵
2021
ハッシュ値生成装置
鍵
ハッシュ関数h
ハッシュ関数h
ハッシュ値生成装置
2011
R=kP
2008
整数倍演算部
圧縮・暗号化
圧縮・暗号化部
2012
2014
圧縮・暗号化
圧縮・暗号化部
2015
暗号文
R
c_1
c_2
・・・
2016
2017
2018
2019

【図２１】

図２１


2102
2103
2104
2105
2123
2106 平文
2107
ベースポイントP
公開鍵Q1
公開鍵Q2
公開鍵Q3
公開鍵Q4
最初のNビット
・・・
2113
q1
q1x
乱数生成
乱数生成装置
k
2114
2115
2116
2124
(x1, y1)=kQ1
(x2, y2)=kQ2
(x3, y3)=kQ3
(x4, y4)=kQ4
x1
x2
x3
x4
整数倍演算部
2126
2127
2128
ハッシュ関数h
h(x2)
h(x3)
h(x4)
ハッシュ値生成装置
2119
h(x1)
2125
しきい値ロジック
2122
鍵
2120
整数倍演算部
R=kP
2123
圧縮・暗号化
圧縮・暗号化部
2121
暗号文
R
f1
f2
c1
2101
データ暗号化装置
2108
2109
2110
2111
2112

【図２２】

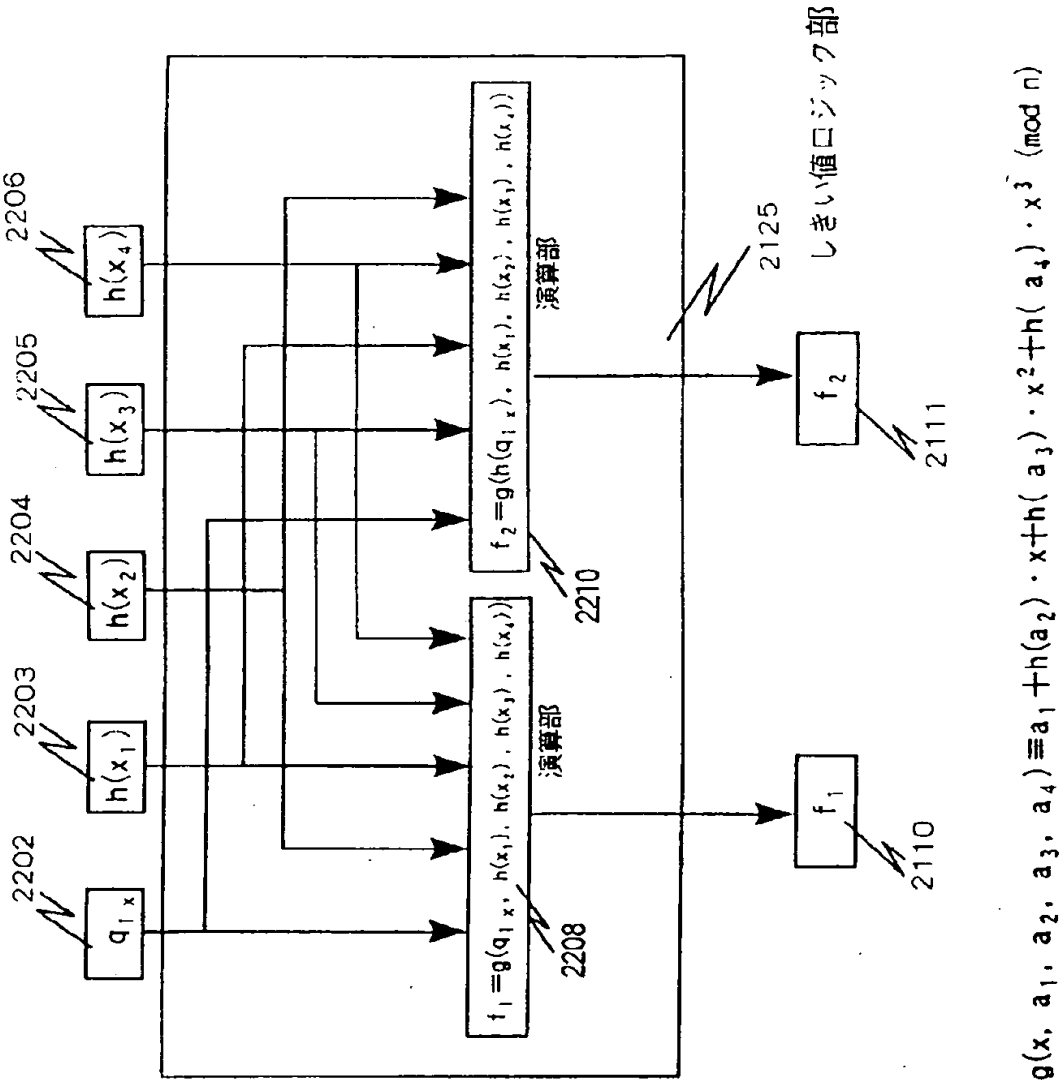
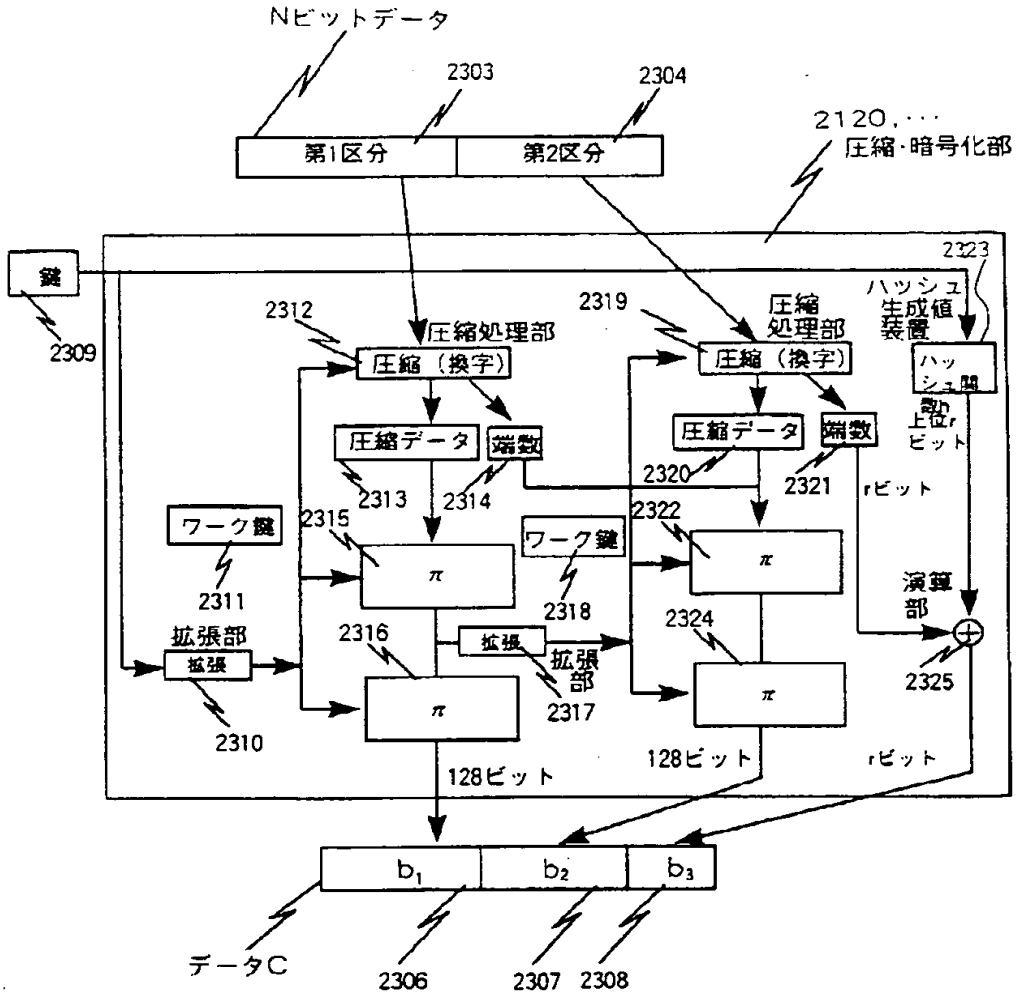

2206
2205
2204
2203
2202
h(x4)
h(x3)
h(x2)
h(x1)
q1x
f2=g(h(q1x), h(x1), h(x2), h(x3), h(x4))
f1=g(q1x, h(x1), h(x2), h(x3), h(x4))
演算部
演算部
2210
2208
2125
しきい値ロジック部
f2
f1
2111
2110
g(x, a1, a2, a3, a4)≡a1+h(a2)・x+h(a3)・x²+h(a4)・x³ (mod n)

図22

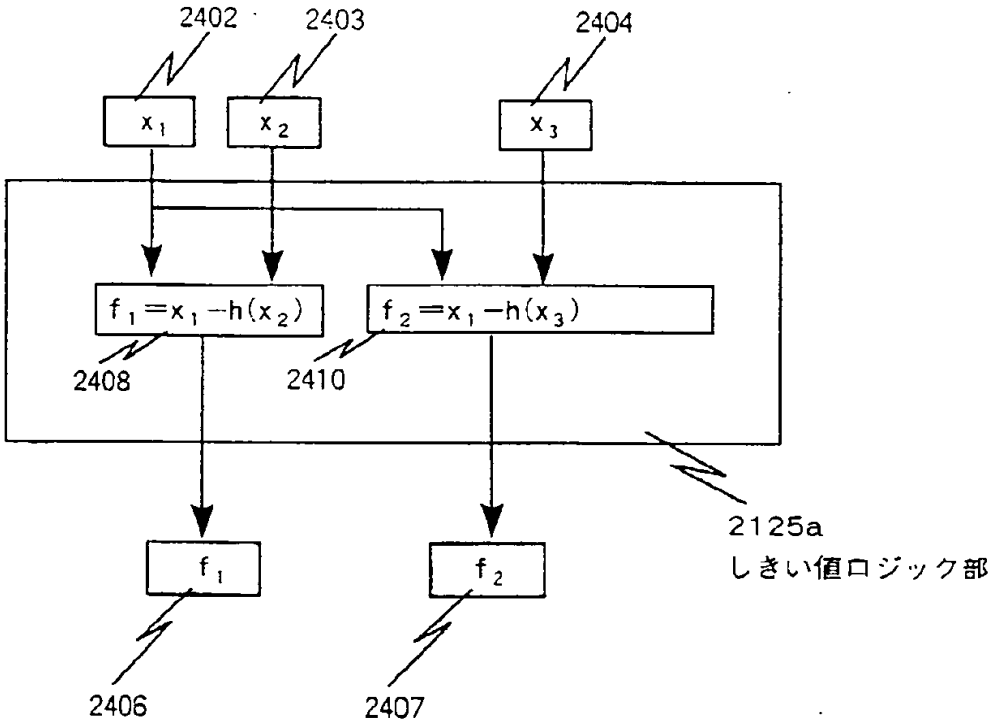
【図２３】

図２３


Nビットデータ
2303
2304
第1区分
第2区分
2120,・・・
圧縮・暗号化部
鍵
2309
2323
ハッシュ生成値装置
ハッシュ関数
上位rビット
2312
圧縮処理部
圧縮（換字）
2319
圧縮処理部
圧縮（換字）
圧縮データ
端数
2313
2314
2320
2321
rビット
ワーク鍵
2311
2315
π
ワーク鍵
2318
2322
π
演算部
2324
2316
拡張部
拡張
拡張部
2317
2325
2310
π
π
128ビット
128ビット
rビット
b1
b2
b3
データC
2306
2307
2308

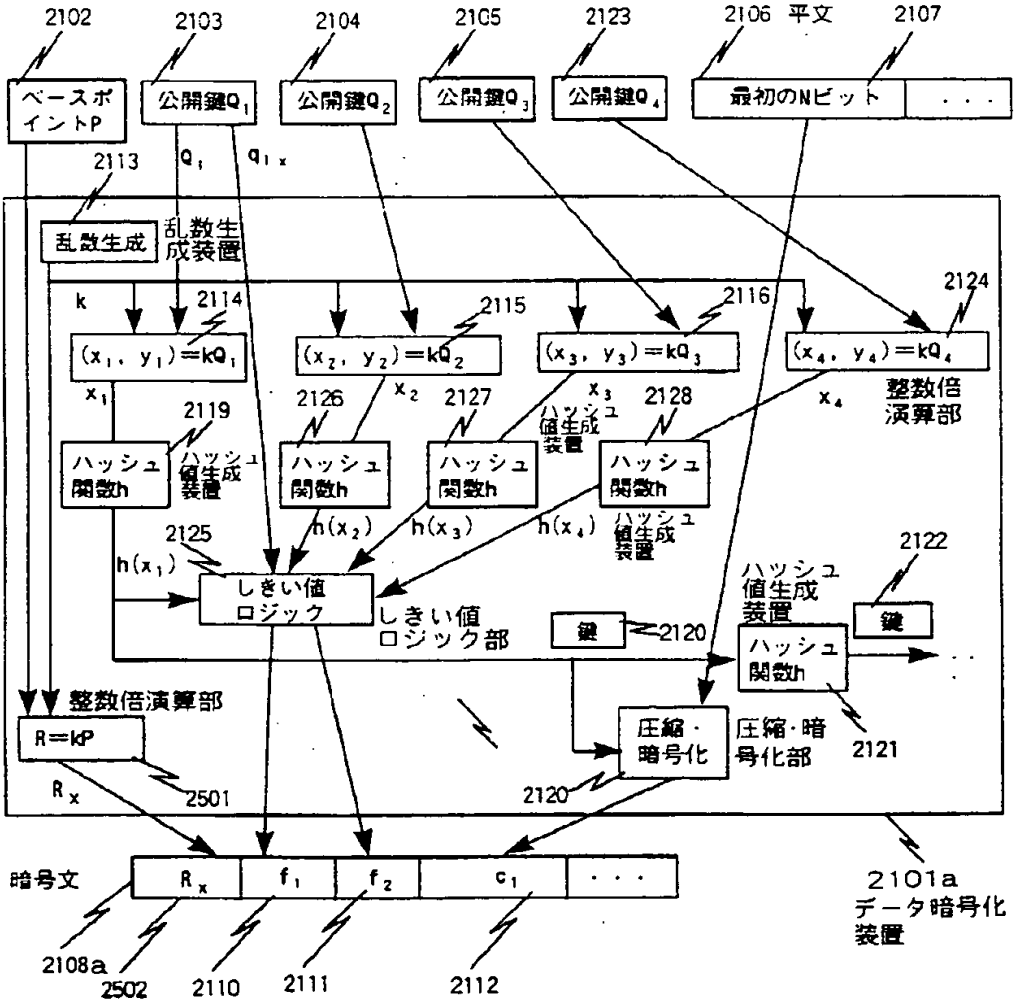
【図２４】

図24


2402
2403
2404
x1
x2
x3
f1=x1−h(x2)
f2=x1−h(x3)
2408
2410
2125a
しきい値ロジック部
f1
f2
2406
2407

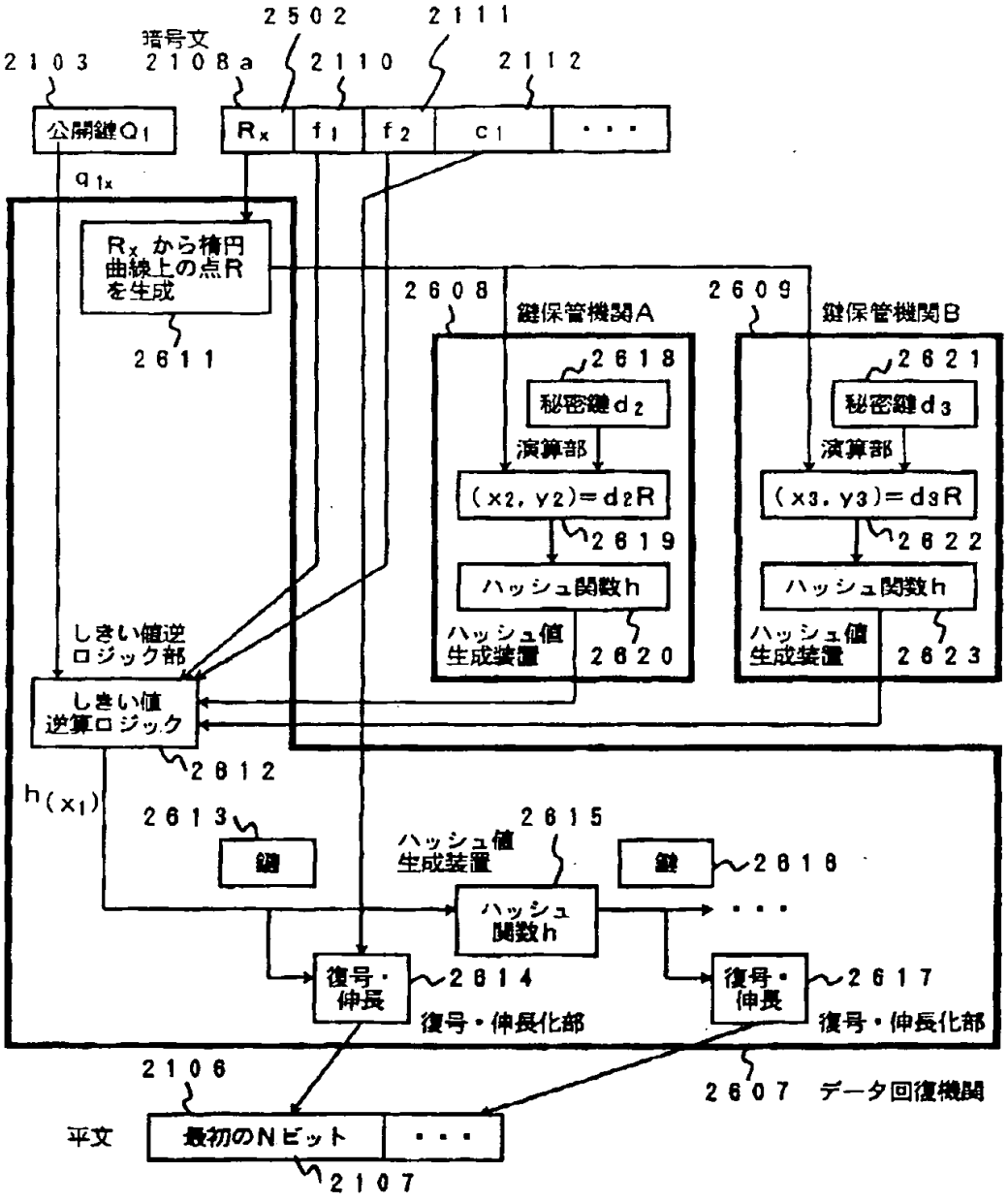
【図２５】

図２５


2102
2103
2104
2105
2123
2106 平文
2107
ベースポイントP
公開鍵Q1
公開鍵Q2
公開鍵Q3
公開鍵Q4
最初のNビット
・・・
2113
乱数生成
乱数生成装置
k
2114
2115
2116
2124
(x1, y1)=kQ1
(x2, y2)=kQ2
(x3, y3)=kQ3
(x4, y4)=kQ4
x1
x2
x3
x4
整数倍演算部
2119
2126
2127
2128
ハッシュ関数h
ハッシュ値生成装置
h(x2)
h(x3)
h(x4)
2125
h(x1)
しきい値ロジック
しきい値ロジック部
鍵
2120
2122
ハッシュ値生成装置
鍵
ハッシュ関数h
2121
整数倍演算部
R=kP
Rx
2501
圧縮・暗号化
圧縮・暗号化部
2120
暗号文
Rx
f1
f2
c1
・・・
2101a
データ暗号化装置
2108a
2502
2110
2111
2112

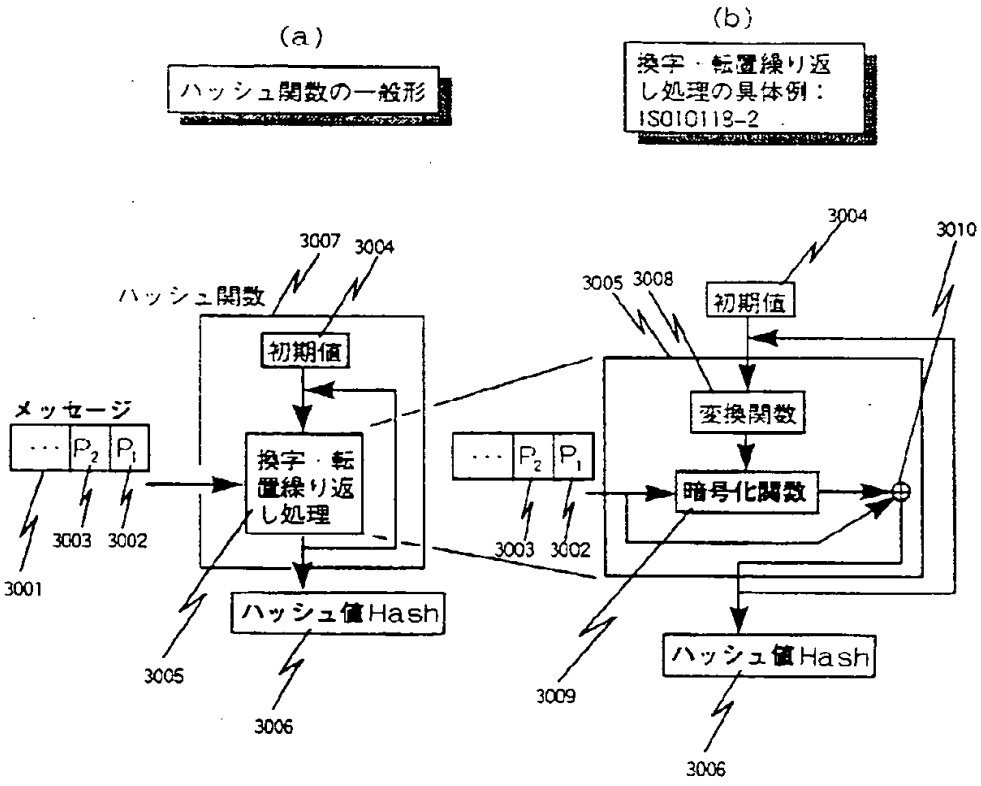
【図２６】

図２６


2502
2111
暗号文
2103
2108a
2110
2112
公開鍵Q1
Rx
f1
f2
c1
・・・
q1x
Rxから楕円曲線上の点Rを生成
2611
2608
鍵保管機関A
2609
鍵保管機関B
2618
秘密鍵d2
演算部
(x2, y2)=d2R
2619
ハッシュ関数h
ハッシュ値生成装置
2620
2621
秘密鍵d3
演算部
(x3, y3)=d3R
2622
ハッシュ関数h
ハッシュ値生成装置
2623
しきい値逆ロジック部
しきい値逆算ロジック
2612
h(x1)
2613
鍵
2615
ハッシュ値生成装置
鍵
2618
ハッシュ関数h
・・・
復号・伸長
2614
復号・伸長化部
復号・伸長
2617
復号・伸長化部
2106
平文
最初のNビット
・・・
2107
2607　データ回復機関

【図２７】

図27


(a)
ハッシュ関数の一般形
(b)
換字・転置繰り返し処理の具体例：ISO10118-2
ハッシュ関数
3007
3004
初期値
メッセージ
・・・ P2 P1
換字・転置繰り返し処理
3001
3003
3002
3005
ハッシュ値Hash
3006
3004
3010
3005
3008
初期値
変換関数
・・・ P2 P1
暗号化関数
3003
3002
3009
ハッシュ値Hash
3006